User's Manual

32

# RH850/P1x-C Group

## User's Manual: Hardware

Renesas microcontroller
RH850 Family

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.1.30 Dec, 2017

# Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.3.0-1 November 2016)

# Notes for CMOS Devices

**(1) Voltage application waveform at input pin:**

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between VIL (MAX) and VIH (MIN) due to noise, etc., the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between VIL (MAX) and VIH (MIN).

**(2) Handling of unused input pins:**

Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to power supply or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.

**(3) Precaution against ESD:**

A strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it when it has occurred. Environmental control must be adequate. When it is dry, a humidifier should be used. It is recommended to avoid using insulators that easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors should be grounded. The operator should be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with mounted semiconductor devices.

**(4) Status before initialization:**

Power-on does not necessarily define the initial status of a MOS device. Immediately after the power source is turned ON, devices with reset functions have not yet been initialized. Hence, power-on does not guarantee output pin levels, I/O settings or contents of registers. A device is not initialized until the reset signal is received. A reset operation must be executed immediately after power-on for devices with reset functions.

**(5) Power ON/OFF sequence:**

In the case of a device that uses different power supplies for the internal operation and external interface, as a rule, switch on the external power supply after switching on the internal power supply. When switching the power supply off, as a rule, switch off the external power supply and then the internal power supply. Use of the reverse power on/off sequences may result in the application of an overvoltage to the internal elements of the device, causing malfunction and degradation of internal elements due to the passage of an abnormal current. The correct power on/off sequence must be judged separately for each device and according to related specifications governing the device.

**(6) Input of signal during power off state:**

Do not input signals or an I/O pull-up power supply while the device is not powered. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.

# How to Use This Manual

**Readers** This manual is intended for users who wish to understand the functions of the RH850/P1x-C and design application systems using the following RH850/P1x-C microcontrollers:

**Purpose** This manual is intended to give users an understanding of the hardware functions of the RH850/P1x-C shown in the *Organization* below.

**Organization** This manual is divided into four parts: Hardware (this manual), Flash Hardware interface (RH850/P1x-C Flash Memory User's Manual: Hardware Interface), Architecture (RH850G3M User's Manual: Software) and Security (RH850/P1x-C User's Manual: Hardware (Security)).

| Hardware | Flash Memory I/F | Software | Security |
|---|---|---|---|
| Pin functions | Module Configuration | Overview | Basic Security |
| CPU function | Address Map | Processor Model | Secure Watchdog Timer |
| On-chip peripheral functions | Registers | Register Reference | ICUMC |
| Flash memory programming | Flash Sequencer Modes | Exceptions and Interrupts | |
| | FACI Command | Memory Management | |
| | Security Functions | Instruction Reference | |
| | Protection Function | Reset | |
| | | Appendix | |

**How to read this manual** It is assumed that the readers of this manual have general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.

To understand the overall functions of the RH850/P1x-C.

→ Read this manual according to the Contents.

To understand the details of programming of Flash

→ See RH850/P1x-C Flash Memory User's Manual: Hardware Interface available separately.

To understand the details of an instruction function

→ See RH850G3M User's Manual: Software available separately.

To understand the details of a security function

→ See RH850/P1x-C User's Manual: Hardware (Security) available separately.

**Conventions** Data significance: Higher digits on the left and lower digits on the right

Active low representation: xxxZ

Memory map address: Higher addresses on the top and lower addresses on the bottom

Note: Footnote for item marked with Note in the text

Caution: Information requiring particular attention

Remark: Supplementary information

Numeric representation: Binary ... xxxx or $xxxx_B$

Decimal ... xxxx

Hexadecimal ... $xxxx_H$

Prefix indicating power of 2 (address space, memory capacity):

K (kilo): $2^{10} = 1,024$

M (mega): $2^{20} = 1,024^2$

G (giga): $2^{30} = 1,024^3$

# Description of Registers

Each register description includes register access, register address, and register value after a reset, a bit chart, illustrating the arrangement of bits, and a table of bits, describing the meaning of the bit settings.

The standard format for bit charts and tables are described below.

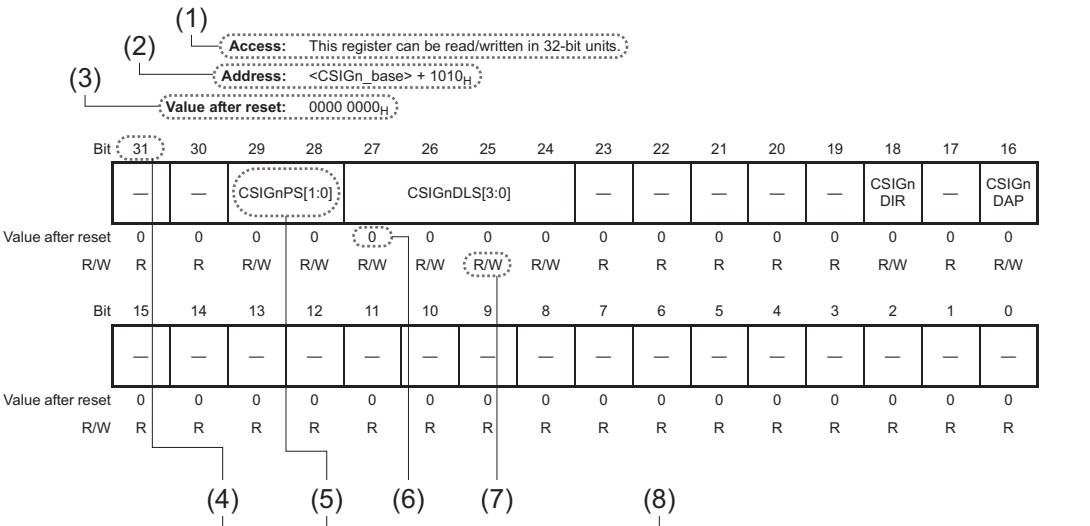



**Table 14.19 CSIGnCFG0 register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | The write value should always be the value after reset. |
| 29, 28 | CSIGnPS[1:0] | Specifies parity. |

| CSIGn PS1 | CSIGn PS0 | Transmission | Reception |
|---|---|---|---|
| 0 | 0 | No parity transmitted | No parity is waited for. |
| 0 | 1 | Add parity bit fixed at 0 | Parity bit is waited for but not judged. |
| 1 | 0 | Add odd parity | Odd parity bit is waited for. |
| 1 | 1 | Add even parity | Even parity bit is waited for. |

| | | |
|---|---|---|
| 27 to 24 | CSIGnDLS [3:0] | Specifies data length.<br>0: Data length is 16 bits<br>1: Data length is 1 bit<br>2: Data length is 2 bits<br>...<br>15: Data length is 15 bits<br>**CAUTION**<br>For a data length of less than 7 bits, do not set bits CSIGnCFG0.CSIGnDLS[3:0] for a value 1 to 6 when the extended data length function is disabled with bit CSIGnCTL1.CSIGnEDLE set to 0.<br>It is forbidden to transmit two consecutive data with a data length of less than 7 bits. |
| 23 to 19 | Reserved | The write value should always be the value after reset. |

(1) Access

The register can be accessed in the bit unit indicated here.

(2) Address

This is the register address.
For base address, see description of base address in each section.

(3) Value after a reset (in hexadecimal notation)

This is the value of all bits of the register after a reset. Values for bytes are given as numbers in the range from 0 to 9 and letters from A to F or as X where they are undefined.

(4) Bit position

This is the bit number.
The bits are numbered from 31 to 0 for 32-bit registers, 15 to 0 for 16-bit registers, and 7 to 0 for 8-bit registers.

(5) Bit name

Bit name or field name is indicated.
When clearly identifying the digits of a bit field is required, do so by using a form such as CSIGnDLS[3:0] above.
Indicate reserved bits by using a dash (—).

(6) Value after a reset (in binary notation)

This is the bit values after a reset.

| | | |
|---|---|---|
| 0 | : | The value after a reset is 0. |
| 1 | : | The value after a reset is 1. |
| — | : | The value after a reset is undefined. |

(7) R/W

This is the bit attribute of all bits of the register.

| | | |
|---|---|---|
| R/W | : | The bit or field is readable and writable. |
| R | : | The bit or field is readable.<br>Note that all reserved bits are indicated as R.<br>When written, the value specified in the bit chart or the value after a reset should be written. |
| W | : | This bit or field is writable.When read, the value is undefined. If a value is indicated in the bit chart, the value is returned. |

(8) Function

This is function of the bit.

All trademarks and registered trademarks are the property of their respective owners.

# Table of Contents


Section 1 Overview ..... 40

1.1 Outline ..... 40
1.2 Product List ..... 41
1.3 Block Diagram ..... 44
1.4 Features ..... 45
1.5 Difference among P1M-C, P1H-C and P1H-CE ..... 57

Section 2 Pin Functions ..... 58

2.1 Pin Connection Diagrams ..... 58
2.1.1 P1M-C (QFP-144) (TOP View) ..... 59
2.1.2 P1M-C (BGA-292) (TOP View) ..... 60
2.1.3 P1H-C (4MB) (BGA-292) (TOP View) ..... 60
2.1.4 P1H-C (8MB) (BGA-292) (TOP View) ..... 61
2.1.5 P1H-CE (BGA-404) (TOP View) ..... 61
2.1.6 P1M-C/P1H-C (BGA-156) (TOP View) ..... 62
2.1.7 Term Definition ..... 62
2.2 Pin List ..... 64
2.2.1 Pin List and Function assignment ..... 64
2.2.2 Pin Status ..... 73
2.2.3 Pin Function assignments ..... 76
2.2.4 Pin Function name ..... 82
2.2.5 Recommended Connection of Unused Pins ..... 96
2.3 Port Overview ..... 99
2.3.1 Introduction ..... 99
2.3.2 Functional Overview ..... 99
2.3.3 Operation Mode ..... 100
2.3.4 Pin data input/output ..... 102
2.4 Port Functions ..... 103
2.4.1 Port Control Logic Block Diagram ..... 103
2.4.2 JTAG port 0 (JP0) ..... 104
2.4.3 Port 0 (P0) ..... 105
2.4.4 Port 1 (P1) ..... 106
2.4.5 Port 2 (P2) ..... 107
2.4.6 Port 3 (P3) ..... 108
2.4.7 Port 4 (P4) ..... 109
2.4.8 Port 5 (P5) ..... 110
2.4.9 Port 6 (P6) ..... 111
2.4.10 Port 7 (P7) ..... 112
2.4.11 Port 8 (P8) ..... 113
2.4.12 Port 9 (P9) ..... 114
2.4.13 Register Address Mapping ..... 115
2.5 Port Register Description ..... 116
2.5.1 Pn/JP0 — Port Register ..... 117
2.5.2 PPRn/JPPR0 — Port Pin Read Register ..... 117

2.5.3 PMn/JPM0 — Port Mode Register ... 118
2.5.4 PMCn/JPMC0 — Port Mode Control Register ... 119
2.5.5 PFCn — Port Function Control Register ... 119
2.5.6 PFCEn/JPFCE0 — Port Function Control Expansion Register ... 120
2.5.7 PNOTn/JPNOT0 — Port Not Register ... 121
2.5.8 PSRn/JPSR0 — Port Set Reset Register ... 122
2.5.9 PMSRn/JPMSR0 — Port Mode Set/Reset Register ... 123
2.5.10 PMCSRn/JPMCSR0 — Port Mode Control Set/Reset Register ... 124
2.5.11 PINVn/JPINV0 — Port output value Inversion Register ... 125
2.5.12 PIBCn/JPIBC0 — Port Input Buffer Control Register ... 126
2.5.13 PBDCn/JPBDC0 — Port Bi-Direction Control Register ... 127
2.5.14 PIPCn — Port IP Control Register ... 128
2.5.15 PUn/JPU0 — Pull-Up Option Register ... 129
2.5.16 PDn/JPD0 — Pull-down option Register ... 130
2.5.17 PODCn/JPODC0 — Port Open Drain Control Register ... 131
2.5.18 PODCEn/JPODCE0 — Port Open Drain Control Expansion Register ... 132
2.5.19 PDSCn/JPDSC0 — Port drive strength control register ... 133
2.5.20 PUCCn/JPUCC0 — Port Universal Characteristic Control Register ... 134
2.5.21 PISn/JPIS0 — Port input buffer selection register ... 135
2.5.22 PCRn_m/JPCR0_m — Port Control Register ... 136
2.6 Port Setting Flow Example ... 138
2.7 Noise Filter & Edge Detector ... 142
2.7.1 Port filter assignment ... 142
2.7.2 Noise Filter Assignment ... 144
2.7.3 Description of Port Noise Filter & Edge/Level Detection ... 149
2.8 Difference among P1M-C, P1H-C and P1H-CE ... 159

Section 3 CPU System ... 160

3.1 Overview ... 160
3.1.1 Block Configuration ... 160
3.2 CPU ... 162
3.2.1 Core Functions ... 162
3.2.2 Instruction Cache and Data Buffer ... 201
3.2.3 Inter-Processor Interrupts ... 204
3.2.4 Reliability Functions ... 204
3.3 Inter CPU Functions Overview ... 235
3.3.1 Processor Element Identifier ... 235
3.3.2 Inter-Processor Interrupt Function ... 235
3.3.3 Exclusive Function ... 235
3.3.4 Write-Through Buffer for Global RAM ... 238
3.4 Usage Notes ... 243
3.4.1 Synchronization of Store Instruction Completion and Subsequent Instruction Generation ... 243
3.4.2 Error response handling ... 244
3.4.3 Speculative accesses handling ... 244

3.4.4 Timing supervision ... 244
3.4.5 Clock Ratio Change ... 244
3.4.6 Register Initialization ... 246
3.4.7 Notes on Prefetch ... 249
3.4.8 System Register Hazards ... 250
3.4.9 Restrictions to use DMAC0/1, DTS ICUM ... 251
3.4.10 Restrictions to use H-Bus Master (Ethernet, HS-USRT, FlexRay) ... 251
3.5 Difference among P1M-C, P1H-C and P1H-CE ... 252

Section 4 Address Map ... 253

4.1 Address Map ... 253
4.2 Address space viewed from each bus master ... 256
4.2.1 Space in which instructions can be fetched ... 256
4.3 Error notification for an access to unmapped area ... 257
4.3.1 Unmapped Code Flash area access error ... 257
4.3.2 Unmapped Global RAM area access error ... 257
4.3.3 Unmapped Local RAM area access error ... 257
4.3.4 Unmapped Local On-chip I/O area access error ... 257
4.3.5 Details of P-Bus area ... 258
4.3.6 Details of H-Bus area ... 269
4.3.7 Unmapped access error from H-bus master ... 270
4.3.8 Debug area ... 270
4.3.9 Other area ... 270
4.3.10 Registers ... 271
4.4 Guard function ... 273
4.5 Difference among P1M-C, P1H-C and P1H-CE ... 273

Section 5 Operating Modes ... 274

5.1 Features ... 274
5.1.1 Normal Operation Mode ... 275
5.1.2 Serial Flash Programming Mode ... 275
5.1.3 Boundary SCAN Mode ... 275
5.2 Input Pins ... 276
5.3 Register Description ... 277
5.3.1 List of Registers ... 277
5.3.2 MODE — Mode Register ... 278

Section 6 Interrupt Controller (INTC) ... 279

6.1 Features ... 279
6.2 RH850/P1x-C Interrupt sources ... 281
6.2.1 RH850/P1x-C Interrupt sources ... 281
6.3 Edge Detection Configuration ... 291
6.4 Interrupt Controller Control Registers ... 292
6.4.1 EICn — EI level interrupt control registers ... 292

6.4.2 IMRm — EI level interrupt mask registers (m = 0 to 7) ..... 294
6.4.3 FNC — FE level NMI status register ..... 295
6.4.4 FIC — FE level maskable interrupt status register ..... 296
6.4.5 FEINTFn — FE level maskable interrupt event status register ..... 297
6.4.6 FEINTFCn — FEINT event clear register ..... 298
6.4.7 FEINTFMSKn — FE level maskable interrupt event mask register ..... 299
6.4.8 EIBD0 to EIBD255 — EI Level Interrupt Bind Registers 0 to 255 ..... 300
6.4.9 SINTRn — Software Interrupt Registers (n = 0 to 2) ..... 303
6.4.10 IPIR_CHn — Inter Processor Interrupt Register ..... 304
6.5 Interrupt Function System Registers ..... 305
6.5.1 ISPR — Priority of interrupt being serviced ..... 305
6.5.2 PMR — Interrupt priority masking ..... 305
6.5.3 ICSR — Interrupt control status ..... 305
6.5.4 INTCFG — Interrupt function setting ..... 305
6.6 Operation When Acknowledging an Interrupt Exception ..... 306
6.7 Return from Interrupts ..... 308
6.8 Interrupt Operation ..... 309
6.8.1 Mask function of EI level maskable interrupt INTn ..... 309
6.8.2 Interrupt priority level judgment ..... 309
6.8.3 Interrupt request acceptance condition and the priority ..... 314
6.8.4 Exception priority of interrupts and the priority mask ..... 314
6.8.5 Interrupt priority mask ..... 314
6.8.6 Priority mask function ..... 314
6.8.7 FEINT Source selection ..... 315
6.8.8 Interrupt management ..... 316
6.9 Exception Handler Address ..... 317
6.9.1 Direct Vector Method ..... 318
6.9.2 Table Reference Method ..... 320
6.10 Broadcast notification function processor elements ..... 322
6.10.1 Example: ..... 322
6.11 Interrupt latency ..... 324
6.12 Difference among P1M-C, P1H-C and P1H-CE ..... 325

Section 7 DMA ..... 332

7.1 Overview ..... 332
7.1.1 Overview ..... 332
7.1.2 Clock Supply ..... 332
7.1.3 Term Definition ..... 333
7.2 DMA Function ..... 334
7.2.1 Basic Operation of DMA Transfer ..... 334
7.2.2 Channel Priority Order ..... 339
7.2.3 Reload Function ..... 342
7.2.4 Chain Function ..... 346
7.2.5 DMAC Operation ..... 348

7.2.6 DTS Operation ..... 354
7.3 Temporarily Suspending DMA transfers ..... 361
7.3.1 Suspension, restart and abortion a DMAC channel ..... 361
7.3.2 Suspension, Resume, and Transfer Abort of a DTS ..... 363
7.3.3 Masking and Clearing a Hardware DMA Transfer Request by the DTFR ..... 364
7.3.4 Masking and Clearing a Hardware DMA Transfer Request by the DTSFSL ..... 364
7.3.5 List of Suspend, Resume, and Transfer Abort Functions ..... 364
7.4 Error Control ..... 365
7.4.1 Type of Error ..... 365
7.4.2 DMA Transfer Error ..... 365
7.4.3 DTSRAM Error ..... 366
7.5 Reliability Function ..... 368
7.5.1 Overview ..... 368
7.5.2 Register Access Protection Function ..... 368
7.5.3 Master Information Inherit Function ..... 369
7.5.4 Other Reliability Functions ..... 370
7.6 Setting Up DMA Transfer ..... 371
7.6.1 Overview of Setting Up DMA ..... 371
7.6.2 Setting Up the Overall DMA Operation ..... 372
7.6.3 Setting Up the DMA Channel Setting ..... 373
7.7 DMA Trigger Source ..... 375
7.7.1 List of DMA Trigger Sources ..... 375
7.8 DTS Trigger Source ..... 379
7.8.1 List of DTS Trigger Sources ..... 379
7.9 Global Register ..... 384
7.9.1 List of Global Register Addresses ..... 384
7.9.2 Details of Global Registers ..... 386
7.10 DMAC Channel Register ..... 412
7.10.1 DMAC Channel Register Address ..... 412
7.10.2 Details of DMAC Channel Registers ..... 413
7.11 DTS Channel Register ..... 431
7.11.1 Transfer information of the DTS (TI) ..... 431
7.11.2 DTS Channel Register Address ..... 435
7.11.3 Details of DTS Channel Registers ..... 436
7.12 DTS Merge Function Register ..... 450
7.12.1 DTS Merge Function Register Descriptions ..... 451
7.13 DTS Trigger Select Registers ..... 467
7.13.1 DTS Trigger Select Register Descriptions ..... 467
7.14 H-Bus DMA Halt Function ..... 471
7.14.1 H-Bus DMA Halt register Address ..... 471
7.14.2 Detail of H-Bus DMA Halt register ..... 471
7.15 Difference among P1M-C, P1H-C and P1H-CE ..... 473

Section 8 Reset Controller ..... 474
8.1 Features ..... 474
8.2 Input/Output Pins ..... 477
8.3 Register Description ..... 477
8.3.1 RESF — Reset Factor Register ..... 480
8.3.2 RESFC — Reset Factor Clear Register ..... 482
8.3.3 ICUMRESF — ICUMC Reset Factor Register ..... 484
8.3.4 ICUMRESFC — ICUMC Reset Factor Clear Register ..... 485
8.3.5 SWSRESA0 — Software System Reset Assertion Register 0 ..... 486
8.3.6 SWSRESA1 — Software System Reset Assertion Register 1 ..... 487
8.3.7 SWARESA0 — Software Application Reset Assertion Register 0 ..... 488
8.3.8 SWARESA1 — Software Application Reset Assertion Register 1 ..... 489
8.3.9 SWLRESA2 — Software Limited Reset Assertion Register for ICUMC ..... 490
8.3.10 SWLRESA3 — Software Limited Reset Assertion Register for MCAN ..... 491
8.3.11 SWLRESA4 — Software Limited Reset Assertion Register for FlexRay ..... 492
8.3.12 SWLRESA5 — Software Limited Reset Assertion Register for GTM ..... 493
8.3.13 SWLRESA6 — Software Limited Reset Assertion Register for Ethernet ..... 494
8.3.14 SWLRESA7 — Software Limited Reset Assertion Register for SENT ..... 495
8.3.15 SWLRESA8 — Software Limited Reset Assertion Register for HS-USRT ..... 496
8.3.16 SWLRESA10 — Software Limited Reset Assertion Register for CSIH ..... 497
8.3.17 SWLRESA11 — Software Limited Reset Assertion Register for RLIN3 ..... 498
8.3.18 SWLRESA12 — Software Limited Reset Assertion Register for AD ..... 499
8.3.19 SWLRESS2 — Software Limited Reset Status Register for ICUMC ..... 500
8.3.20 SWLRESS3 — Software Limited Reset Status Register for MCAN ..... 501
8.3.21 SWLRESS4 — Software Limited Reset Status Register for FlexRay ..... 502
8.3.22 SWLRESS5 — Software Limited Reset Status Register for GTM ..... 503
8.3.23 SWLRESS6 — Software Limited Reset Status Register for Ethernet ..... 504
8.3.24 SWLRESS7 — Software Limited Reset Status Register for SENT ..... 505
8.3.25 SWLRESS8 — Software Limited Reset Status Register for HS-USRT ..... 506
8.3.26 SWLRESS10 — Software Limited Reset Status Register for CSIH ..... 507
8.3.27 SWLRESS11 — Software Limited Reset Status Register for RLIN3 ..... 508
8.3.28 SWLRESS12 — Software Limited Reset Status Register for AD ..... 509
8.3.29 RESC — Reset Configuration Register ..... 510
8.3.30 BOOTCTRL — Boot Control Register ..... 511
8.3.31 STAC_DTSRAM — RAM Initialization Mode Control Register for DTS_RAM ..... 512
8.3.32 STAC_GRAM — RAM Initialization Mode Control Register for GRAM ..... 513
8.3.33 STAC_LM0 — RAM Initialization Mode Control Register for PE1 ..... 514
8.3.34 STAC_LM1 — RAM Initialization Mode Control Register for PE2 ..... 515
8.3.35 STAC_LM2 — RAM Initialization Mode Control Register for ICUMC ..... 516
8.3.36 STAC_LM3 — RAM Initialization Mode Control Register for MCAN ..... 517
8.3.37 STAC_LM4 — RAM Initialization Mode Control Register for FlexRay ..... 518
8.3.38 STAC_LM5 — RAM Initialization Mode Control Register for GTM ..... 519
8.3.39 STAC_LM10 — RAM Initialization Mode Control Register for CSIH ..... 520
8.4 Operation ..... 521
8.4.1 Reset Categories ..... 521

8.4.2 Reset Sources ..... 522
8.4.3 Reset Flags ..... 524
8.4.4 Read Configuration Data from FLASH ..... 524
8.4.5 HW BIST ..... 524
8.4.6 RAM initialization ..... 524
8.4.7 Start Up of Cores ..... 525
8.4.8 Reset Mask function ..... 525
8.4.9 Reset Output (RESETOUTZ) ..... 525

Section 9 Power Supply ..... 526

9.1 Features ..... 526
9.2 External Pin List ..... 526
9.3 Block Diagram ..... 528
9.3.1 P1M-C (Dual power supply) ..... 528
9.3.2 P1H-C and P1H-CE (Dual power supply) ..... 529
9.4 Connection Example ..... 530
9.4.1 Dual Power Supply ..... 530
9.5 Power Up/Down Timing ..... 531

Section 10 Core Voltage Monitor (CVM) ..... 532

10.1 Overview of RH850/P1x-C CVM ..... 532
10.1.1 Units ..... 532
10.1.2 Register Addresses ..... 532
10.1.3 Clock Supply ..... 532
10.1.4 Input/Output Pins ..... 532
10.2 Overview ..... 533
10.2.1 Block Diagram ..... 534
10.3 Registers ..... 535
10.3.1 List of Registers ..... 535
10.3.2 Reset of Registers ..... 536
10.3.3 CVMF — CVM factor register ..... 537
10.3.4 CVMFC — CVMF clear register ..... 538
10.3.5 CVMDE — CVM detection enable register ..... 539
10.3.6 CVMDEW — CVMDE write register ..... 540
10.3.7 CVMDMASK — CVM detection output diagnosis mask register ..... 541
10.3.8 CVMDIAG — CVM DIAG mode setting register ..... 542
10.3.9 CVMMON — CVM Monitor register ..... 543
10.4 Operation ..... 544
10.4.1 CVM basic function ..... 544
10.4.2 CVM function in FBIST and Serial programing mode (Mask of CVMOUTZ pin and CVM reset) ..... 547
10.4.3 CVM diagnosis function ..... 547
10.5 Usage Notes ..... 549

Section 11 Temperature Sensor ..... 550
11.1 Overview of RH850/P1x-C Temperature Sensor ..... 550
11.1.1 Channels ..... 550
11.1.2 Register Address ..... 550
11.1.3 Clock Supply ..... 550
11.1.4 Interrupt Request ..... 551
11.1.5 Hardware Reset ..... 551
11.1.6 External Input/Output Signals ..... 551
11.2 Overview ..... 552
11.3 Configuration ..... 553
11.3.1 List of Registers ..... 554
11.4 Details of Registers ..... 555
11.4.1 OTS0OTSTCR — Temperature Measurement Start Control Register ..... 555
11.4.2 OTS0OTENDCR — Temperature Measurement End Control Register ..... 555
11.4.3 OTS0OTCR — Temperature Sensor Control Register ..... 556
11.4.4 OTS0OTFCR — Temperature Sensor Flag Clear Register ..... 557
11.4.5 OTS0OTFR — Temperature Sensor Flag Register ..... 558
11.4.6 OTS0OTSTR — Temperature Status Register ..... 559
11.4.7 OTS0OTDR — Temperature Data Register ..... 561
11.4.8 OTS0HTBRmn — High-Temperature Border mn Register ..... 562
11.4.9 OTS0LTBRAn — Low-Temperature Border An Register ..... 563
11.4.10 OTS0TDLR — Temperature Difference Limiting Register ..... 564
11.4.11 OTS0COEFFRn — Coefficient n Register ..... 565
11.5 Operation ..... 566
11.5.1 Functional description ..... 566
11.5.2 Examples of Temperature Measurement Operation ..... 566
11.5.3 Temperature Measurement End Interrupt Request ..... 567
11.5.4 Temperature Alarm Error and Temperature Rise /Drop Interrupt and Temperature Sensor Error Interrupt Requests ..... 568
Section 12 Clock Controller ..... 569
12.1 Features ..... 569
12.1.1 External Input / Output pins ..... 569
12.2 Overview ..... 570
12.2.1 Type of clocks ..... 570
12.2.2 Block Diagram ..... 571
12.2.3 Clock sources ..... 572
12.2.4 PLL for the System Clock ..... 572
12.2.5 PLL Configuration ..... 572
12.2.6 Main Oscillator ..... 574
12.2.7 Internal Oscillator ..... 574
12.3 Register Description ..... 575
12.3.1 Writing to protected registers ..... 575
12.3.2 Register Overview ..... 575

12.3.3 CLKD0DIV — Clock divider 0 divisor register ... 577
12.3.4 CLKD0STAT — Clock divider 0 status register ... 579
12.3.5 CLKD1DIV — Clock divider 1 divisor register ... 580
12.3.6 CLKD1STAT — Clock divider 1 status register ... 582
12.3.7 CLKD2DIV — Clock divider 2 divisor register ... 583
12.3.8 CLKD2STAT — Clock divider 2 status register ... 584
12.3.9 CLKD3DIV — Clock divider 3 divisor register ... 585
12.3.10 CLKD3STAT — Clock divider 3 status register ... 586
12.3.11 CKSC0C — Clock selector 0 control register ... 587
12.3.12 CKSC0S — Clock selector 0 status register ... 588
12.3.13 CKSC2C — Clock selector 2 control register ... 589
12.3.14 CKSC2S — Clock selector 2 status register ... 590
12.3.15 CKSC3C — Clock selector 3 control register ... 591
12.3.16 CKSC3S — Clock selector 3 status register ... 592
12.3.17 CKSC4C — Clock selector 4 control register ... 593
12.3.18 CKSC4S — Clock selector 4 status register ... 594
12.4 Operation ... 595
12.4.1 Clock Selection for Operating Frequency ... 595
12.4.2 External Clock Output Pins (ECLK0/1) ... 596
12.5 Usage Notes ... 597
12.5.1 How to Connect a Crystal Resonator ... 597
12.6 Difference among P1M-C, P1H-C and P1H-CE ... 598

Section 13 Clock Monitor (CLMA) ... 599
13.1 Features of RH850/P1x-C CLMA ... 599
13.1.1 Number of Channels ... 599
13.1.2 Register Base Addresses ... 599
13.1.3 Clock Supply ... 600
13.1.4 Reset Sources ... 600
13.1.5 Internal Input/Output Signals ... 601
13.2 Overview ... 602
13.2.1 Functional Overview ... 602
13.3 Enabling CLMA ... 603
13.4 Functions ... 603
13.4.1 Detection of Abnormal Clock Frequencies ... 603
13.4.2 Notification of Abnormal Clock Frequency ... 605
13.4.3 Self-Test ... 606
13.5 Registers ... 607
13.5.1 Register Protection ... 607
13.5.2 List of Registers ... 607
13.5.3 Reset of Registers ... 608
13.5.4 CLMAnCTL0 — CLMAn Control Register 0 ... 610
13.5.5 CLMAnCMPL — CLMAn comparison register L ... 611
13.5.6 CLMAnCMPH — CLMAn comparison register H ... 611

13.5.7 CLMAnPCMD — CLMAn Protection command register ... 612
13.5.8 CLMAnPS — CLMAn Protection status register ... 612
13.5.9 CLMAnEMU0 — CLMAn Emulation Register 0 ... 613
13.5.10 CLMATEST — Clock Monitor Test register ... 614
13.5.11 CLMATESTS — Clock Monitor Test Status register ... 616
13.5.12 CLMA5CTL1 — CLMA5 control register 1 ... 617
13.6 Operation ... 618
13.6.1 Procedures to enable CLMAn ... 618
13.6.2 Procedures to Reset by CLMATEST.RESCLM ... 618
13.7 Difference among P1M-C, P1H-C and P1H-CE ... 619

Section 14 Power Down Modes ... 620
14.1 Features ... 620
14.2 Power down Modes ... 620
14.2.1 HALT Mode ... 620
14.2.2 Module standby ... 620
14.3 Register Description ... 622
14.3.1 List of Registers ... 622
14.3.2 MSR_LM2 — Module Standby Register for ICUMC ... 623
14.3.3 MSR_LM3 — Module Standby Register for MCAN ... 624
14.3.4 MSR_LM4 — Module Standby Register for FlexRay ... 625
14.3.5 MSR_LM5 — Module Standby Register for GTM ... 626
14.3.6 MSR_LM6 — Module Standby Register for Ethernet ... 627
14.3.7 MSR_LM7 — Module Standby Register for SENT ... 628
14.3.8 MSR_LM8 — Module Standby Register for HS-USRT ... 629
14.3.9 MSR_LM10 — Module Standby Register for CSI-H ... 630
14.3.10 MSR_LM11 — Module Standby Register for RLIN3 ... 631
14.3.11 MSR_LM12 — Module Standby Register for AD ... 632

Section 15 External Memory Controller (MEMC) ... 633
15.1 External Memory Controller (MEMC) Overview ... 633
15.1.1 Number of Channels ... 633
15.1.2 Register Addresses ... 633
15.1.3 Clock Supply ... 634
15.1.4 External Input/Output Signals ... 634
15.2 Overview ... 635
15.2.1 Functional Overview ... 635
15.2.2 Block Diagram ... 635
15.3 Registers ... 636
15.3.1 MEMCBCT0 — Bus cycle type configuration register ... 637
15.3.2 MEMCDWC0 — Data wait configuration register ... 638
15.3.3 MEMCDHC — Data hold wait configuration register ... 639
15.3.4 MEMCICC0 — Idle cycle configuration register ... 640
15.4 Operation ... 641

15.4.1 Chip Select Output Function ... 641
15.4.2 Operation Enable/Operation Disable Setting Function ... 642
15.4.3 Wait functions ... 642
15.4.4 Programmable Data Wait Function ... 642
15.4.5 Data Hold Wait Function ... 643
15.4.6 Idle Insertion Function ... 644
15.4.7 Memory access timing examples ... 645
15.5 Difference among P1M-C, P1H-C and P1H-CE ... 646

Section 16 Clocked Serial Interface H (CSIH) ... 647

16.1 Features of RH850/P1x-C CSIH ... 647
16.1.1 Units ... 647
16.1.2 Register Base Address ... 648
16.1.3 Clock Supply ... 648
16.1.4 Interrupt Requests ... 649
16.1.5 Hardware Reset ... 649
16.1.6 External Input/Output Signals ... 650
16.1.7 Data Consistency Check ... 651
16.2 Overview ... 652
16.2.1 Functional Overview ... 652
16.2.2 Functional Overview Description ... 653
16.2.3 Block Diagram ... 654
16.3 Registers ... 655
16.3.1 List of Registers ... 655
16.3.2 CSIHnCTL0 — CSIH control register 0 ... 657
16.3.3 CSIHnCTL1 — CSIH control register 1 ... 658
16.3.4 CSIHnCTL2 — CSIH control register 2 ... 661
16.3.5 CSIHnMCTL1 — CSIH Memory control register 1 ... 663
16.3.6 CSIHnMCTL2 — CSIH Memory control register 2 ... 664
16.3.7 CSIHnTX0W — CSIH transmit data register 0 for word access ... 666
16.3.8 CSIHnTX0H — CSIH transmit data register 0 for half word access ... 668
16.3.9 CSIHnRX0W — CSIH receive data register 0 for word access ... 669
16.3.10 CSIHnRX0H — CSIH receive data register 0 for half word access ... 670
16.3.11 CSIHnMRWP0 — CSIH memory read/write pointer register 0 ... 671
16.3.12 CSIHnSTR0 — CSIH status register 0 ... 673
16.3.13 CSIHnSTCR0 — CSIH status clear register 0 ... 678
16.3.14 CSIHnMCTL0 — CSIH Memory control register 0 ... 679
16.3.15 CSIHnCFGx — CSIH Configuration register (x = 0 to 7) ... 680
16.3.16 CSIHnBRSx — CSIH baud rate setting register (x = 0 to 3) ... 685
16.3.17 PMMAnCTL — PMM control register ... 686
16.3.18 PMMAnTCTLx — PMM TG control register (x = 0 to 7) ... 687
16.3.19 PMMAnTTXm — PMM TG transmit buffer (x = 0 to 7) ... 689
16.3.20 PMMAnRX — PMM TG receive buffer ... 690
16.3.21 PMMAnTCNTx — PMM TGx counter (x = 0 to 7) ... 691
16.3.22 PMMAnSTR0 — PMM TG status register 0 ... 692

16.3.23 PMMAnSTR1 — PMM TG status register 1 ... 693
16.3.24 PMMAnSTC — PMM TG status clear register ... 694
16.3.25 List of Caution ... 695
16.4 Interrupt Sources ... 698
16.4.1 Overview ... 698
16.4.2 General interrupt delay ... 699
16.4.3 CSIHTIC (communication interrupt) ... 699
16.5 Operation ... 707
16.5.1 Operating modes (master/slave) ... 707
16.5.2 Master/slave connections ... 709
16.5.3 Chip selection (CS) features ... 711
16.5.4 Chip select timing details ... 714
16.5.5 Transmission clock selection ... 718
16.5.6 CSIH buffer memory ... 720
16.5.7 Data transfer modes ... 722
16.5.8 Data length selection ... 723
16.5.9 Serial data direction selection ... 726
16.5.10 SS Function ... 727
16.5.11 Handshake function ... 729
16.5.12 Error detection ... 732
16.5.13 Loop-back mode ... 741
16.5.14 Enforced chip select idle setting ... 742
16.5.15 Idle state control mode ... 743
16.5.16 Silent mode ... 746
16.5.17 Limited Reset and Module stand-by ... 746
16.5.18 Chip select Return to Inactive ... 747
16.5.19 Select the Timing of Outputting INT_CSIHTIC ... 749
16.6 Operating Procedures ... 751
16.6.1 Procedures in direct access mode ... 751
16.6.2 Procedures in transmit-only buffer mode ... 755
16.6.3 Procedures in dual buffer mode ... 759
16.6.4 Procedures in FIFO mode ... 765
16.7 Priority Management Module (PMM) ... 769
16.7.1 Transfer groups ... 769
16.7.2 Block Diagram ... 776
16.7.3 Functional Overview ... 777
16.7.4 PMM operation ... 781
16.7.5 PMM operation description ... 788
16.8 Difference among P1M-C, P1H-C and P1H-CE ... 791

Section 17 High Speed USRT (HS-USRT) ... 792

17.1 RH850/P1x-C HS-USRT Overview ... 792
17.1.1 Number of Channels ... 792
17.1.2 Register Addresses ... 792
17.1.3 Clock Supply ... 793

17.1.4 Interrupts and DMA/DTS ... 793
17.1.5 HS-USRT Hardware Reset ... 794
17.1.6 Input/Output Signals ... 794
17.2 Overview ... 796
17.2.1 Block Diagram ... 797
17.3 Registers ... 798
17.3.1 HSUSnCKEN — HS-USRT Clock Enable Register ... 799
17.3.2 HSUSnSRST — HS-USRT Software Reset Register ... 800
17.3.3 HSUSnEN — HS-USRT Communication Enable Register ... 801
17.3.4 HSUSnMD — HS-USRT Mode Register ... 802
17.3.5 HSUSnCTL — HS-USRT Communication Control Register ... 803
17.3.6 HSUSnTDIV — HS-USRT Transmit Clock Division Select Register ... 804
17.3.7 HSUSnTSAR — HS-USRT Transmit Data Source Address Register ... 805
17.3.8 HSUSnTPFM — HS-USRT Transmit Frame Number Register ... 806
17.3.9 HSUSnRDAR — HS-USRT Data Destination Address Register ... 807
17.3.10 HSUSnRPFM — HS-USRT Receive Frame number Register ... 808
17.3.11 HSUSnRGFM — HS-USRT Receive Group Frame Number Register ... 809
17.3.12 HSUSnIREN — HS-USRT Interrupt Enable Register ... 810
17.3.13 HSUSnIRST — HS-USRT Interrupt Status Register ... 812
17.3.14 HSUSnIRCL — HS-USRT Interrupt Status Clear Register ... 815
17.3.15 HSUSnCST — HS-USRT Communication Status Register ... 817
17.3.16 HSUSnCTFM — HS-USRT Current Transmit Frame Number Register ... 818
17.3.17 HSUSnCRFM — HS-USRT Current Receive Frame Number Register ... 819
17.4 Operation ... 820
17.4.1 Overview of HS-USRT Operation ... 820
17.4.2 Control of the HS-USRT Pins ... 822
17.4.3 Data Format ... 823
17.4.4 Buffer Configuration ... 826
17.4.5 Interrupt Requests ... 828
17.4.6 Discarding Received Data ... 831
17.4.7 Operation for Reception of Overrun Frames ... 832
17.4.8 Timeout ... 832
17.4.9 Setting Procedure ... 833
17.4.10 Notes ... 838
17.4.11 Limited Reset and Module stand-by ... 838
17.4.12 Data Storage and Address Space ... 839
17.5 Difference among P1M-C, P1H-C and P1H-CE ... 840

Section 18 LIN/UART Interface (RLIN3) ... 841

18.1 Features of RH850/P1x-C RLIN3 ... 841
18.1.1 Units and Channels ... 841
18.1.2 Register Base Address ... 842
18.1.3 Clock Supply ... 842
18.1.4 Interrupt Request ... 843
18.1.5 Reset sources ... 843

18.1.6 External Input/Output Signals ... 844
18.2 Overview ... 845
18.2.1 Functional Overview ... 845
18.2.2 Block Diagram ... 849
18.2.3 Description of Blocks ... 849
18.3 Registers ... 850
18.3.1 List of Registers ... 850
18.3.2 LIN Master Related Registers ... 852
18.3.3 LIN Slave Related Registers ... 878
18.3.4 UART Related Registers ... 902
18.4 Interrupt Sources ... 927
18.5 Operation ... 928
18.5.1 Modes ... 928
18.5.2 LIN Reset Mode ... 930
18.5.3 LIN Mode ... 931
18.5.4 UART Mode ... 956
18.5.5 LIN Self-Test Mode ... 971
18.5.6 Baud Rate Generator ... 979
18.5.7 Noise Filter ... 982
18.5.8 Limited Reset and Module stand-by ... 983
18.6 Difference among P1M-C, P1H-C and P1H-CE ... 984

Section 19 CAN Controller (MCAN) ... 985

19.1 Features ... 985
19.1.1 Number of Units and Channels ... 986
19.1.2 Register Base Address ... 986
19.1.3 Clock Supply ... 987
19.1.4 RAM areas for MCAN ... 987
19.1.5 Interrupt Requests ... 988
19.1.6 External Input / Output Pins ... 988
19.2 Overview ... 989
19.2.1 Functional overview ... 989
19.2.2 Block Diagram ... 991
19.3 Registers ... 992
19.3.1 List of Registers ... 992
19.4 Operation ... 993
19.4.1 Procedure of Module Standby and Limited Reset ... 993
19.5 MCAN ... 995
19.5.1 Overview ... 995
19.5.2 Programmer’s Model ... 998
19.5.3 Functional Description ... 1065
19.6 M_TTCAN ... 1090
19.6.1 Overview ... 1090
19.6.2 Programmer’s Model ... 1095

19.6.3 Functional Description ..... 1186
19.6.4 TTCAN Operation ..... 1212
19.7 Usage Notes ..... 1237
19.7.1 Port sharing between M_CAN IP and RLIN ..... 1237
19.8 CAUTION ..... 1237
19.9 Difference among P1M-C, P1H-C and P1H-CE ..... 1240

Section 20 FlexRay (FLX) ..... 1241

20.1 Overview of RH850/P1x-C FlexRay ..... 1241
20.1.1 Channels ..... 1241
20.1.2 Register Base Addresses ..... 1241
20.1.3 Clock Supply ..... 1242
20.1.4 Interrupts ..... 1242
20.1.5 FlexRay Hardware Reset ..... 1243
20.1.6 I/O Signals ..... 1243
20.1.7 Functions ..... 1244
20.1.8 Block Diagram ..... 1245
20.2 Register ..... 1248
20.2.1 Register Map ..... 1248
20.2.2 FlexRay Operation register ..... 1253
20.2.3 Special Registers ..... 1259
20.2.4 Interrupt Registers ..... 1260
20.2.5 FlexRay Timer Registers ..... 1284
20.2.6 CC Control Registers ..... 1293
20.2.7 CC Status Registers ..... 1328
20.2.8 Message Buffer Control Registers ..... 1353
20.2.9 Message Buffer Status Registers ..... 1361
20.2.10 Identification Registers ..... 1374
20.2.11 Input Buffer ..... 1375
20.2.12 Output Buffer ..... 1386
20.2.13 Data Transfer Control Register ..... 1404
20.2.14 Data Transfer Status Register ..... 1417
20.2.15 NTU loss monitor register ..... 1432
20.3 Operation ..... 1447
20.3.1 FlexRay Module Operation Control ..... 1447
20.3.2 Communication Cycle ..... 1449
20.3.3 Communication Modes ..... 1452
20.3.4 Clock Synchronization ..... 1453
20.3.5 Error Handling ..... 1455
20.3.6 Communication Controller States ..... 1457
20.3.7 Network Management ..... 1471
20.3.8 Filtering and Masking ..... 1472
20.3.9 Transmit Process ..... 1475
20.3.10 Receive Process ..... 1477
20.3.11 FIFO Function ..... 1479

20.3.12 Message Handling ..... 1481
20.3.13 Message RAM ..... 1490
20.3.14 Module Interrupt ..... 1498
20.3.15 Assignment of FlexRay Configuration Parameters ..... 1499
20.3.16 Usage of Data Transfer ..... 1501
20.3.17 Byte Alignment ..... 1525
20.3.18 FlexRay NTU supervision ..... 1527
20.3.19 Port sharing between FlexRay IP and RLIN ..... 1530
20.3.20 Wakeup capability ..... 1530
20.3.21 Limited Reset and Module stand-by ..... 1530
20.4 CAUTION ..... 1531
20.5 Difference among P1M-C, P1H-C and P1H-CE ..... 1533

Section 21 Ethernet Controller (ETNA) ..... 1534

21.1 Feature Overview of RH850/P1x-C Ethernet Controller ..... 1534
21.1.1 Number of Channels and Ports ..... 1534
21.1.2 Register Addresses ..... 1534
21.1.3 Clock Supply ..... 1535
21.1.4 Interrupts and DMA/DTS ..... 1535
21.1.5 Difference table of P1M-C, P1H-C, P1H-CE product ..... 1536
21.1.6 External Input/Output Signals ..... 1536
21.2 Overview ..... 1537
21.2.1 Functional Overview ..... 1537
21.2.2 Block Diagram ..... 1538
21.3 Registers ..... 1539
21.3.1 Overview of Register ..... 1539
21.3.2 ETNAnHDMMDRm — HDMAC Operation Mode Register m ..... 1548
21.3.3 ETNAnHDMISRm — HDMAC Interrupt Status Register m ..... 1549
21.3.4 ETNAnHDMIERm — HDMAC Interrupt Enable Register m ..... 1550
21.3.5 ETNAnARSTR — Software Reset Register ..... 1551
21.3.6 ETNAnECMRm — E-MAC Mode Register m ..... 1552
21.3.7 ETNAnECSRm — MAC Status Register m ..... 1555
21.3.8 ETNAnECSIPRm — E-MAC Interrupt Permission Register m ..... 1556
21.3.9 ETNAnPIRm — PHY Interface Register m ..... 1557
21.3.10 ETNAnPLSRm — PHY link status Register m ..... 1558
21.3.11 ETNAnPIPRm — PHY interrupt polarity register m ..... 1559
21.3.12 ETNAnMAHRm — MAC Address High Register m ..... 1560
21.3.13 ETNAnMALRm — MAC Address Low Register m ..... 1561
21.3.14 ETNAnRFLRm — Receive Frame Length Register m ..... 1562
21.3.15 ETNAnTROCRm — Transmit Retry Over Counter Register m ..... 1563
21.3.16 ETNAnCDCRm — Delayed Collision Detect Counter Register m ..... 1564
21.3.17 ETNAnLCCRm — Lost Carrier Counter Register m ..... 1565
21.3.18 ETNAnCEFCRm — CRC Error Frame Receive Counter Register m ..... 1566
21.3.19 ETNAnFRECRm — Frame Receive Error Counter Register m ..... 1567
21.3.20 ETNAnTSFRCRm — Too-Short Frame Receive Counter Register m ..... 1568

21.3.21 ETNAnTLFRCRm — Too-Long Frame Receive Counter Register m ..... 1569
21.3.22 ETNAnRFCRm — Residual-Bit Frame Receive Counter Register m ..... 1570
21.3.23 ETNAnMAFCRm — Multicast Address Frame Receive Counter Register m..... 1571
21.3.24 ETNAnAPRm — Automatic PAUSE Frame Register m ..... 1572
21.3.25 ETNAnMPRm — Manual PAUSE Frame Register m..... 1573
21.3.26 ETNAnTPAUSERm — Automatic PAUSE Frame Retransmit Count Register m ..... 1574
21.3.27 ETNAnTPFRCRm — Automatic PAUSE Frame Retransmit Counter Register m..... 1575
21.3.28 ETNAnPFTCRm — PAUSE Frame Transmit Counter Register m..... 1576
21.3.29 ETNAnPFRCRm — PAUSE Frame Receive Counter Register m ..... 1577
21.3.30 ETNAnEMRm — ETNA Mode Register m ..... 1578
21.3.31 ETNAnBCFRRm — Broadcast Frame Receive Count Set Register m ..... 1579
21.3.32 ETNAnCRBCFRRm — Count Reset of BCFRR m ..... 1580
21.3.33 ETNAnTSU_CTRST — TSU Counter Reset Register ..... 1581
21.3.34 ETNAnTSU_FWSLm — Receive Function Set Register..... 1582
21.3.35 ETNAnTSU_FWSLC — TSU Function Set Register (Common)..... 1583
21.3.36 ETNAnTSU_FWSR — TSU Status Register..... 1584
21.3.37 ETNAnTSU_FWINMK — TSU Status Interrupt Mask Register..... 1585
21.3.38 ETNAnTSU_VTAGm — VLANtag Set Register ..... 1587
21.3.39 ETNAnTSU_ADSBSY — CAM Entry Table Busy Register..... 1588
21.3.40 ETNAnTSU_TEN — CAM Entry Table Enable Register ..... 1589
21.3.41 ETNAnTSU_POST1 — CAM Entry Table POST1 Register..... 1592
21.3.42 ETNAnTSU_POST2 — CAM Entry Table POST2 Register..... 1594
21.3.43 ETNAnTSU_POST3 — CAM Entry Table POST3 Register..... 1596
21.3.44 ETNAnTSU_POST4 — CAM Entry Table POST4 Register..... 1598
21.3.45 ETNAnTSU_ADRH0 to ETNAnTSU_ADRH31 — CAM Entry Table 0H to 31H Registers ..... 1600
21.3.46 ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31 — CAM Entry Table 0L to 31L Registers ..... 1601
21.3.47 ETNAnTXNLCRm — Transmit Frame Counter Register (Port m) (Normal Transmission Only)..... 1602
21.3.48 ETNAnTXALCRm — Transmit Frame Counter Register (Port m) (Normal and Erroneous Transmission)..... 1603
21.3.49 ETNAnRXNLCRm — Receive Frame Counter Register (Port m) (Normal Reception Only)..... 1604
21.3.50 ETNAnRXALCRm — Receive Frame Counter Register (Port m) (Normal and Erroneous Reception)..... 1605
21.3.51 ETNAnEDSRm — E-DMAC Start Register m ..... 1606
21.3.52 ETNAnEDMRm — E-DMAC Mode Register m ..... 1607
21.3.53 ETNAnEDTRRm — E-DMACm Transmit Request Register m ..... 1609
21.3.54 ETNAnEDRRRm — E-DMACm Receive Request Register m..... 1610
21.3.55 ETNAnTDLARm — Transmit Descriptor List Start Address Register m ..... 1611
21.3.56 ETNAnRDLARm — Receive Descriptor List Start Address Register m ..... 1612
21.3.57 ETNAnEESRm — E-MAC/E-DMAC Status Register m ..... 1613
21.3.58 ETNAnEESIPRm — E-MAC/E-DMAC Status Interrupt Permission Register m..... 1616
21.3.59 ETNAnTRSCERm — Transmit/Receive Status Copy Enable Register m ..... 1618
21.3.60 ETNAnRMFCRm — Receive Missed-Frame Counter Register m ..... 1620
21.3.61 ETNAnFDRm — FIFO Depth Register m..... 1621
21.3.62 ETNAnRMCRm — Receiving Method Control Register m..... 1622

21.3.63 ETNAnRDFARm — Receive Descriptor Fetch Address Register m ... 1623
21.3.64 ETNAnRDFXRm — Receive Descriptor Finished Address Register m ... 1624
21.3.65 ETNAnRDFFRm — Receive Descriptor Final Flag Register m ... 1625
21.3.66 ETNAnTDFARm — Transmit Descriptor Fetch Address Register m ... 1626
21.3.67 ETNAnTDFXRm — Transmit Descriptor Finished Address Register m ... 1627
21.3.68 ETNAnTDFFRm — Transmit Descriptor Final Flag Register m ... 1628
21.3.69 ETNAnFCFTRm — Overflow Alert FIFO Threshold Register m ... 1629
21.3.70 ETNAnTFSRm — Transmit FIFO Status Register m ... 1630
21.3.71 ETNAnRFSRm — Receive FIFO Status Register m ... 1631
21.3.72 ETNAnRPADIRm — Receive Data Padding Insert Register m ... 1632
21.4 Operation ... 1633
21.4.1 Limited Reset and Module Stop ... 1633
21.4.2 Basic Frame ... 1633
21.4.3 PAUSE Control Frame ... 1634
21.4.4 Magic Packet™ ... 1634
21.4.5 Transmission ... 1635
21.4.6 Transmission Clock ... 1635
21.4.7 Collision Detection and Retransmission ... 1635
21.4.8 CRC Function ... 1635
21.4.9 Transmission Frame Data Padding Function ... 1635
21.4.10 Full-/half-duplex Communication ... 1635
21.4.11 Inter-packet Gap (IPG) ... 1635
21.4.12 Reception ... 1636
21.4.13 Reception Clock ... 1636
21.4.14 Detection of Preamble and SFD ... 1636
21.4.15 CRC Function ... 1636
21.4.16 Promiscuous Mode ... 1636
21.4.17 Magic Packet Detection ... 1636
21.4.18 Flow Control ... 1636
21.4.19 PAUSE Frame Transmission ... 1637
21.4.20 PAUSE Frame Reception ... 1637
21.4.21 Loopback Function ... 1637
21.4.22 MAC Management Counters ... 1638
21.4.23 ETNA Operation ... 1639
21.4.24 Descriptors and Descriptor List ... 1641
21.4.25 Transmit Descriptor ... 1642
21.4.26 Receive Descriptor ... 1646
21.4.27 Descriptor and Transmit/Receive Buffer ... 1652
21.4.28 Descriptor Pointer ... 1654
21.4.29 Transmission Procedure and Processing Flow ... 1656
21.4.30 Transmission Error Processing ... 1659
21.4.31 Reception Procedure and Processing Flow ... 1659
21.4.32 Reception Error Processing ... 1662
21.4.33 CAM Function ... 1664
21.4.34 Transmit/Receive Processing of Multi-Buffer Frame (Single-Frame/Multi-Descriptor) ... 1665
21.4.35 Padding Insertion in Receive Data ... 1667

21.4.36 Interrupt Sources ... 1668
21.4.37 Activation Procedure ... 1671
21.4.38 Termination Procedure ... 1672
21.4.39 Flow Control ... 1673
21.4.40 Automatic PAUSE Frame Transmission ... 1673
21.4.41 Manual PAUSE Frame Transmission ... 1673
21.4.42 PAUSE Frame Reception ... 1673
21.4.43 0-Time PAUSE Frame Control ... 1673
21.4.44 Back-Pressure Flow Control ... 1674
21.4.45 Magic Packet Detection ... 1674
21.5 Connection to PHY-LSI ... 1675
21.5.1 Accessing MII Registers ... 1675
21.5.2 MII Management Frame Format ... 1675
21.5.3 MII Register Access Procedure ... 1676
21.6 Usage Notes ... 1678
21.6.1 Checksum Calculation of Ethernet Frames ... 1678
21.7 Configuration of MII/RMII Mode ... 1679

Section 22 Single Edge Nibble Transmission (SENT) ... 1680

22.1 RH850/P1x-C SENT Overview ... 1680
22.1.1 Number of Channels ... 1680
22.1.2 Register Addresses ... 1680
22.1.3 Clock Supply ... 1681
22.1.4 Interrupts and DMA requests ... 1681
22.1.5 RSENT Hardware Reset ... 1682
22.1.6 External Input/Output Signals ... 1683
22.2 Overview ... 1684
22.2.1 Block Diagram ... 1685
22.3 Registers ... 1686
22.3.1 RSENTnTSPC — RSENT Timestamp Register ... 1687
22.3.2 RSENTnTSC — RSENT Timestamp Counter ... 1689
22.3.3 RSENTnCC — RSENT Communication Configuration Register ... 1690
22.3.4 RSENTnBRP — RSENT Baud Rate Prescaler Register ... 1693
22.3.5 RSENTnIDE — RSENT Interrupt/DMA Enable Register ... 1695
22.3.6 RSENTnMDC — RSENT Mode Control Register ... 1698
22.3.7 RSENTnSPCT — RSENT SPC Transmission Register ... 1699
22.3.8 RSENTnMST — RSENT Mode Status Register ... 1700
22.3.9 RSENTnCS — RSENT Communication Status Register ... 1702
22.3.10 RSENTnCSC — RSENT Communication Status Clear Register ... 1706
22.3.11 RSENTnSRTS — RSENT Slow Channel Receive Timestamp Register ... 1709
22.3.12 RSENTnSRXD — RSENT Slow Channel Receive Data Register ... 1710
22.3.13 RSENTnCPL — RSENT Calibration Pulse Length Register ... 1712
22.3.14 RSENTnML — RSENT Message Length Register ... 1713
22.3.15 RSENTnFRTS — RSENT Fast Channel Receive Timestamp Register ... 1714
22.3.16 RSENTnFRXD — RSENT Fast Channel Receive Data Register ... 1715

22.3.17 RSENT0TSSEL — RSENT0 Timestamp Mode Selection Register ..... 1717
22.3.18 RSENT1TSSEL — RSENT1 Timestamp Mode Selection Register ..... 1718
22.3.19 RSENT2TSSEL — RSENT2 Timestamp Mode Selection Register ..... 1719
22.3.20 RSENT3TSSEL — RSENT3 Timestamp Mode Selection Register ..... 1720
22.3.21 RSENT4TSSEL — RSENT4 Timestamp Mode Selection Register ..... 1721
22.3.22 RSENT5TSSEL — RSENT5 Timestamp Mode Selection Register ..... 1722
22.3.23 RSENT6TSSEL — RSENT6 Timestamp Mode Selection Register ..... 1723
22.3.24 RSENT7TSSEL — RSENT7 Timestamp Mode Selection Register ..... 1724
22.3.25 RSENT8TSSEL — RSENT8 Timestamp Mode Selection Register ..... 1725
22.3.26 RSENT9TSSEL — RSENT9 Timestamp Mode Selection Register ..... 1726
22.4 Operation ..... 1727
22.4.1 Modes of Operation ..... 1727
22.4.2 Clock Configuration ..... 1730
22.4.3 RSENT Operation ..... 1734
22.4.4 SPC Function ..... 1750
22.4.5 Interrupts and Checks ..... 1752
22.4.6 Limited Reset and Module stand-by ..... 1753
22.5 Difference among P1M-C, P1H-C and P1H-CE ..... 1754

Section 23 Window Watchdog Timer A (WDTA) ..... 1755

23.1 Features ..... 1755
23.1.1 Number of Units and Channels ..... 1755
23.1.2 Register Base Address ..... 1755
23.1.3 Clock Supply ..... 1755
23.1.4 Interrupt Request ..... 1756
23.1.5 External Input / Output Pins ..... 1756
23.2 Overview ..... 1757
23.2.1 Functional Overview ..... 1757
23.2.2 Block Diagram ..... 1758
23.3 Registers ..... 1759
23.3.1 List of Registers ..... 1759
23.3.2 WDTAnWDTE — WDTA enable register ..... 1760
23.3.3 WDTAnMD — WDTA mode register ..... 1761
23.3.4 WDTAnEVAC — WDTA enable VAC register ..... 1762
23.3.5 WDTAnREF — WDTA reference value register ..... 1763
23.4 Operation ..... 1764
23.4.1 WDTA Start-up Options ..... 1764
23.4.2 WDTA after reset release ..... 1765
23.4.3 WDTA trigger ..... 1768
23.4.4 Error detection ..... 1769
23.4.5 75% interrupt request signals ..... 1770
23.4.6 Window function ..... 1771
23.5 Cautions ..... 1772
23.6 Difference among P1M-C, P1H-C and P1H-CE ..... 1772

Section 24 System Timer (STM) ..... 1773
24.1 Features ..... 1773
24.1.1 Number of Units and Channels ..... 1773
24.1.2 Register Base Address ..... 1773
24.1.3 Clock Supply ..... 1774
24.1.4 Interrupt Request ..... 1774
24.1.5 External Input / Output Pins ..... 1774
24.2 Overview ..... 1775
24.2.1 Functional overview ..... 1775
24.2.2 Counter channels and compare registers ..... 1775
24.2.3 Block Diagram ..... 1776
24.3 Registers ..... 1777
24.3.1 List of Registers ..... 1777
24.3.2 STMnCKSEL — STMn timer counter source clock selection register ..... 1778
24.3.3 STMnTS — STMn timer counter start register ..... 1779
24.3.4 STMnTT — STMn timer counter stop register ..... 1780
24.3.5 STMnCSTR — STMn timer counter status register ..... 1781
24.3.6 STMnSTR — STMn timer status register ..... 1782
24.3.7 STMnSTC — STMn timer status clear register ..... 1783
24.3.8 STMnIS — STMn timer interrupt selection register ..... 1784
24.3.9 STMnRM — STMn timer SW reset mask register ..... 1785
24.3.10 STMnCNT0L — STMn timer counter register low (lower 32-bit) ..... 1786
24.3.11 STMnCNT0H — STMn timer counter register high (upper 32-bit) ..... 1787
24.3.12 STMnCMP0lL — STMn timer compare match register low (lower 32-bit) ..... 1788
24.3.13 STMnCMP0lH — STMn timer compare match register high (upper 32-bit) ..... 1789
24.3.14 STMnCNTm — STMn timer counter register ..... 1790
24.3.15 STMnCMPml — STMn timer compare match register ..... 1791
24.4 Operation ..... 1792
24.4.1 Interrupt assignment ..... 1792
24.4.2 Cautions ..... 1793
24.4.3 Attention in using ..... 1795
24.4.4 Access to register ..... 1800
24.4.5 Resets ..... 1801
24.5 Difference among P1M-C, P1H-C and P1H-CE ..... 1802
Section 25 Generic Timer Module (GTM) ..... 1803
25.1 Feature ..... 1803
25.1.1 IP version ..... 1803
25.1.2 Number of Sub-Units and Channels ..... 1803
25.1.3 Register/RAM Base Address ..... 1804
25.1.4 Clock Supply ..... 1804
25.1.5 Interrupt and DMA/DTS Requests ..... 1805
25.1.6 External Input and Output Pins ..... 1807
25.2 Overview ..... 1808

25.2.1 Functional Overview ..... 1808
25.2.2 Block Diagram ..... 1809
25.3 Registers ..... 1810
25.3.1 List of Registers ..... 1810
25.4 Operation ..... 1847
25.4.1 Peripheral Bus Access ..... 1847
25.4.2 Reset ..... 1847
25.4.3 Interrupt Factors ..... 1847
25.4.4 Connection to ADCF ..... 1847
25.4.5 Connection from ADCF ..... 1848
25.4.6 Sub-Module ..... 1848
25.4.7 Safety Mechanism ..... 1848
25.4.8 Procedure of Module Standby and Limited Reset ..... 1849
25.5 Introduction ..... 1850
25.5.1 Overview ..... 1850
25.5.2 Document Structure ..... 1851
25.6 GTM Architecture ..... 1851
25.6.1 Overview ..... 1851
25.6.2 GTM-IP Interfaces ..... 1855
25.6.3 ARU Routing Concept ..... 1858
25.6.4 GTM-IP Clock and Time Base Management (CTBM) ..... 1861
25.6.5 GTM-IP Interrupt Concept ..... 1863
25.6.6 GTM-IP Software Debugger Support ..... 1871
25.6.7 GTM-IP Programming conventions ..... 1871
25.6.8 GTM-IP TOP-Level Configuration Registers Overview ..... 1872
25.6.9 GTM-IP TOP-Level Configuration Registers Description ..... 1873
25.7 Advanced Routing Unit (ARU) ..... 1891
25.7.1 Overview ..... 1891
25.7.2 Special Data Sources ..... 1891
25.7.3 ARU Access via AEI ..... 1891
25.7.4 ARU Interrupt Signals ..... 1893
25.7.5 ARU Configuration Registers Overview ..... 1893
25.7.6 ARU Configuration Registers Description ..... 1894
25.8 Clock Management Unit (CMU) ..... 1908
25.8.1 Overview ..... 1908
25.8.2 Global Clock Divider ..... 1909
25.8.3 Configurable Clock Generation Subunit (CFGU) ..... 1909
25.8.4 Wave Form of Generated Clock Signal CMU_CLK[x] ..... 1910
25.8.5 CMU Configuration Registers Overview ..... 1910
25.8.6 CMU Configuration Register Description ..... 1911
25.9 Time Base Unit (TBU) ..... 1916
25.9.1 Overview ..... 1916
25.9.2 TBU Time Base Channels ..... 1917
25.9.3 TBU Configuration Registers Overview ..... 1917

25.9.4 TBU Registers description ... 1918
25.10 Timer Input Module (TIM) ... 1923
25.10.1 Overview ... 1923
25.10.2 TIM Filter Functionality (FLT) ... 1927
25.10.3 Timeout Detection Unit (TDU) ... 1935
25.10.4 TIM Channel Architecture ... 1937
25.10.5 TIM Interrupt Signals ... 1947
25.10.6 TIM Configuration Registers Overview ... 1947
25.10.7 TIM Configuration Registers Description ... 1948
25.11 ARU-connected Timer Output Module (ATOM) ... 1975
25.11.1 Overview ... 1975
25.11.2 ATOM Channel architecture ... 1980
25.11.3 ATOM Channel modes ... 1983
25.11.4 ATOM Interrupt signals ... 2024
25.11.5 ATOM Register overview ... 2024
25.11.6 ATOM Register description ... 2025
25.12 Dead Time Module (DTM) ... 2051
25.12.1 Overview ... 2051
25.12.2 DTM Channel ... 2053
25.12.3 Phase Shift Control Unit ... 2058
25.12.4 Multiple output signal combination ... 2060
25.12.5 Synchronous update of channel control register 2 ... 2062
25.12.6 DTM output shut off ... 2063
25.12.7 Configuration Register Overview ... 2064
25.12.8 Configuration Register Description ... 2065
25.13 Multi Channel Sequencer (MCS) ... 2078
25.13.1 Overview ... 2078
25.13.2 Architecture ... 2079
25.13.3 Scheduling ... 2081
25.13.4 Memory Organization ... 2086
25.13.5 Instruction Set ... 2086
25.13.6 MCS Internal Register Overview ... 2111
25.13.7 MCS Internal Register Description ... 2111
25.13.8 MCS Configuration Registers Overview ... 2125
25.13.9 MCS Configuration Register Description ... 2126
25.14 Memory Configuration (MCFG) ... 2154
25.14.1 Overview ... 2154
25.14.2 MCFG Configuration Registers Overview ... 2156
25.14.3 MCFG Configuration Registers ... 2157
25.15 Interrupt Concentrator Module (ICM) ... 2159
25.15.1 Overview ... 2159
25.15.2 Bundling ... 2159
25.15.3 ICM Interrupt Signals ... 2161
25.15.4 ICM Configuration Registers Overview ... 2161
25.15.5 ICM Configuration Registers Description ... 2162

25.16 Output Compare Unit (CMP) ..... 2181
25.16.1 Overview ..... 2181
25.16.2 Bitwise Compare Unit (BWC) ..... 2181
25.16.3 Configuration of the Compare Unit ..... 2182
25.16.4 Error Generator ..... 2182
25.16.5 CMP Interrupt Signal ..... 2182
25.16.6 CMP Configuration Registers Overview ..... 2182
25.16.7 CMP Configuration Registers Description ..... 2183
25.17 Monitor Unit (MON) ..... 2192
25.17.1 Overview ..... 2192
25.17.2 Clock Monitoring ..... 2193
25.17.3 CMP error Monitoring ..... 2193
25.17.4 Checking the Characteristics of Signals by MCS ..... 2193
25.17.5 Checking ARU Cycle Time ..... 2194
25.17.6 MON Interrupt Signals ..... 2194
25.17.7 MON Registers Overview ..... 2194
25.17.8 MON Configuration Registers Description ..... 2195
25.18 GTM Device 207 ..... 2206
25.18.1 Architecture Block Diagram ..... 2206
25.18.2 GTM Device 207 Configuration ..... 2207
25.18.3 Register reset values device depended hardcoded ..... 2208
25.18.4 Register reset values defined by hardware configuration ..... 2209
25.18.5 Memory Address Ranges ..... 2216
25.18.6 ARU ..... 2217
25.18.7 ATOM ..... 2219
25.18.8 GTM Application constraints ..... 2219
25.19 GTM Device 208 ..... 2220
25.19.1 Architecture Block Diagram ..... 2220
25.19.2 GTM Device 208 Configuration ..... 2221
25.19.3 Register reset values device depended hardcoded ..... 2222
25.19.4 Register reset values defined by hardware configuration ..... 2223
25.19.5 Memory Address Ranges ..... 2230
25.19.6 ARU ..... 2231
25.19.7 ATOM ..... 2234
25.19.8 GTM Application constraints ..... 2234
25.20 CAUTION ..... 2235
25.21 Difference among P1M-C, P1H-C and P1H-CE ..... 2239

Section 26 Peripheral Interconnect (PIC) ..... 2240

26.1 Feature ..... 2240
26.1.1 Number of Units and Channels ..... 2240
26.1.2 Register Base Address ..... 2240
26.1.3 Clock Supply ..... 2240
26.1.4 Interrupt and DMA/DTS Requests ..... 2240
26.1.5 External Input and Output Pins ..... 2241

26.2 Overview ... 2242
26.2.1 Functional Overview ... 2242
26.2.2 Block Diagram ... 2242
26.3 Registers ... 2243
26.3.1 List of Registers ... 2243
26.3.2 PIC2CTRGPREMUXn — A/D converter trigger select n (n = 0/1) ... 2244
26.3.3 PIC2CTRGMUXn — A/D converter trigger select control n (n = 0/1) ... 2247
26.3.4 PIC2CEDGSELn — A/D converter trigger edge control n (n = 0/1) ... 2248
26.3.5 PIC2CENP2TIM0 — Path to TIM0 enable control register ... 2249
26.3.6 PIC2CENP2TIM1 — Path to TIM1 enable control register ... 2251
26.3.7 PIC2CENHIZDTM — Hi-Z function for GTM output enable control register ... 2253
26.4 Operation ... 2254
26.4.1 ADCF trigger select function ... 2254
26.4.2 ADCF conversion interrupt routed to GTM input ... 2256
26.4.3 Baud rate measurement for an UART (RLIN3) ... 2258
26.4.4 Hi-Z control function over external pin for GTM output ... 2258
26.4.5 GTM output monitor for PWM diagnostic ... 2259
26.5 Difference among P1M-C, P1H-C and P1H-CE ... 2260

Section 27 A/D Converter (ADCF) ... 2261
27.1 Features of RH850/P1x-C ADCF ... 2261
27.1.1 Number of Units ... 2261
27.1.2 Register Base Address ... 2261
27.1.3 Clock Supply ... 2262
27.1.4 Interrupts and DMA ... 2262
27.1.5 Hardware Reset ... 2263
27.1.6 External Input/Output Signals ... 2263
27.1.7 Analog Channels ... 2264
27.1.8 Virtual Channel ... 2264
27.2 Overview ... 2265
27.2.1 Functional Overview ... 2265
27.2.2 Block Diagram ... 2267
27.2.3 Physical Channels, Virtual Channels and Scan Groups ... 2269
27.3 Registers ... 2271
27.3.1 List of Registers ... 2271
27.3.2 ADCF0ADSYNSTCR — AD Synchronization Start Control Register ... 2273
27.3.3 ADCF0ADTSYNSTCR — AD Timer Synchronization Start Control Register ... 2273
27.3.4 ADCFnVCRj — Virtual Channel Register j ... 2274
27.3.5 ADCFnDRj — Data Register j ... 2276
27.3.6 ADCFnDIRj — Data Supplementary Information Register j ... 2278
27.3.7 ADCFnADHALTR — AD Halt Register ... 2279
27.3.8 ADCFnADCR1 — AD Control Register 1 ... 2280
27.3.9 ADCFnMPXCURCR — MPX Current Control Register ... 2281
27.3.10 ADCFnMPXCURR — MPX Current Register ... 2282
27.3.11 ADCFnMPXOWR — MPX Optional Wait Register ... 2283

27.3.12 ADCFnADCR2 — AD Control Register 2 ..... 2284
27.3.13 ADCFnADENDP0 — A/D Conversion Monitor Virtual Channel Pointer ..... 2285
27.3.14 ADCFnSFTCR — Safety Control Register ..... 2286
27.3.15 ADCFnTDCR —Pin Level Diagnostic Control Register ..... 2287
27.3.16 ADCFnULLMTBR0 to 2 — Upper-Limit/Lower-Limit Table Registers 0 to 2 ..... 2288
27.3.17 ADCFnECR — Error Clear Register ..... 2289
27.3.18 ADCFnULER — Upper-Limit/Lower-Limit Error Register ..... 2290
27.3.19 ADCFnOWER — Overwrite Error Register ..... 2291
27.3.20 ADCFnPER — Parity Error Register ..... 2292
27.3.21 ADCFnIDER — ID Error Register ..... 2293
27.3.22 ADCFnSGSTCRx — Scan Group x Start Control Register ..... 2294
27.3.23 ADCFnADTSTCRy — AD Timer y Start Control Register ..... 2295
27.3.24 ADCFnADTENDCRy — AD Timer y End Control Register ..... 2296
27.3.25 ADCFnSGCRx — Scan Group x Control Register ..... 2297
27.3.26 ADCFnSGVCSPx — Scan Group x Start Virtual Channel Pointer ..... 2299
27.3.27 ADCFnSGVCEPx — Scan Group x End Virtual Channel Pointer ..... 2300
27.3.28 ADCFnSGMCYCRx — Scan Group x Multicycle Register ..... 2301
27.3.29 ADCFnSGSRx — Scan Group x Status Register ..... 2302
27.3.30 ADCFnADTIPRy — AD Timer Initial Phase Register y ..... 2304
27.3.31 ADCFnADTPRRy — AD Timer Period Register y ..... 2305
27.3.32 ADCFnULLMSRx — Scan Group x Upper-Limit/Lower-Limit Table Select Register ..... 2306
27.4 Function ..... 2307
27.4.1 Examples of Normal AD Conversion Operation ..... 2307
27.4.2 Example of Normal A/D Conversion Operation in Addition Mode ..... 2309
27.4.3 Example of Operation of External Analog Multiplexer ..... 2310
27.4.4 Example of Synchronous Suspend and Resume Operation ..... 2312
27.4.5 Example of Asynchronous Suspend and Resume Operation ..... 2313
27.4.6 Example of A/D Timer Operation ..... 2314
27.4.7 Diagnostic Function ..... 2315
27.4.8 ADC conversion time ..... 2318
27.4.9 Analog Input Sampling and Scan Group Processing Time ..... 2318
27.4.10 Hardware Trigger Functions ..... 2319
27.4.11 Starting a Scan Group by Using a Hardware Trigger ..... 2320
27.4.12 Starting a Scan Group by Using an A/D Timer Trigger ..... 2320
27.4.13 Starting A/D Timer by Using a Hardware Trigger ..... 2320
27.4.14 Monitoring Function by Using the A/D Conversion Monitor Pin ..... 2320
27.4.15 Scan End Interrupt Request ..... 2321
27.4.16 MPX Interrupt Request ..... 2322
27.4.17 A/D Error Interrupt Request and A/D Parity Error Notification ..... 2323
27.5 Operation ..... 2324
27.5.1 Initial Settings ..... 2324
27.5.2 Trigger Input Flow ..... 2325
27.5.3 Terminating Procedure ..... 2327
27.5.4 Limited Reset and Module stand-by ..... 2327
27.6 Definition of A/D Conversion Accuracy ..... 2328
27.7 Difference among P1M-C, P1H-C and P1H-CE ..... 2329

Section 28 Functional Safety ..... 2330
28.1 Overview ..... 2330
28.2 ECC and EDC ..... 2332
28.2.1 Overview ..... 2332
28.2.2 Code Flash ECC and Address Parity ..... 2334
28.2.3 Data Flash ECC ..... 2348
28.2.4 Local RAM (CPU1/CPU2) ECC ..... 2359
28.2.5 Global RAM ECC ..... 2378
28.2.6 Instruction Cache ECC and EDC ..... 2401
28.2.7 DTS RAM ECC ..... 2423
28.2.8 ECC for Peripheral RAM (32 Bits) ..... 2423
28.2.9 Safety Mechanism on Data Transfer Path ..... 2439
28.2.10 ECC Function ..... 2511
28.3 Lockstep ..... 2520
28.3.1 List of Registers ..... 2520
28.3.2 Details of Registers ..... 2521
28.3.3 Usage Notes ..... 2524
28.4 Memory Protection ..... 2525
28.4.1 Overview ..... 2525
28.4.2 GRG ..... 2532
28.4.3 PBG ..... 2541
28.4.4 HBG ..... 2573
28.5 BIST ..... 2582
28.5.1 List of Registers ..... 2582
28.5.2 Details of Registers ..... 2583
28.6 ECM ..... 2603
28.7 CVM ..... 2603
28.8 CLMA ..... 2603
28.9 WDTA ..... 2603
28.10 DCRB ..... 2603
28.11 Difference among P1M-C, P1H-C and P1H-CE ..... 2603
Section 29 Error Control Module (ECM) ..... 2604
29.1 Features of RH850/P1x-C Group ECM ..... 2604
29.1.1 Units ..... 2604
29.1.2 Register Base Address ..... 2604
29.1.3 Clock Supply ..... 2605
29.1.4 Interrupt Requests ..... 2605
29.1.5 External Output Signals ..... 2605
29.2 Overview ..... 2606
29.2.1 Specification Overview ..... 2606
29.3 Block diagram ..... 2607
29.3.1 Error Input ..... 2608

29.3.2 Operations for Error Output ..... 2618
29.3.3 ERROROUTZ behavior at reset ..... 2618
29.3.4 Error Status ..... 2619
29.3.5 Writing to Protected Registers ..... 2619
29.3.6 Timeout Function for Interrupt Processing ..... 2620
29.3.7 Configuration lock ..... 2620
29.3.8 Masking of “Error clear trigger register” ..... 2621
29.4 Register Specification ..... 2622
29.4.1 List of Registers ..... 2622
29.4.2 ECMmnESET — ECM Master/Checker n Error Set Trigger Register (m = M/C) ..... 2627
29.4.3 ECMmnECLR — ECM Master/Checker n Error Clear Trigger Register (m = M/C) ..... 2628
29.4.4 ECMmnESSTR0 — ECM Master/Checker n Error Source Status Register 0 (m = M/C) ..... 2630
29.4.5 ECMmnESSTR1 — ECM Master/Checker n Error Source Status Register 1 (m = M/C) ..... 2631
29.4.6 ECMmnESSTR2 — ECM Master/Checker n Error Source Status Register 2 (m = M/C) ..... 2632
29.4.7 ECMmnPCMD0 — ECM Master/Checker n Protection Command Register (m = M/C) ..... 2633
29.4.8 ECMnEPCFG — ECM n Error Pulse Configuration Register ..... 2634
29.4.9 ECMnMICFG0 — ECM n Maskable Interrupt Configuration Register 0 ..... 2635
29.4.10 ECMnMICFG1 — ECM n Maskable Interrupt Configuration Register 1 ..... 2636
29.4.11 ECMnMICFG2 — ECM n Maskable Interrupt Configuration Register 2 ..... 2637
29.4.12 ECMnNMICFG0 — ECM n Non-maskable Interrupt Configuration Register 0 ..... 2638
29.4.13 ECMnNMICFG1 — ECM n Non-maskable Interrupt Configuration Register 1 ..... 2639
29.4.14 ECMnNMICFG2 — ECM n Non-maskable Interrupt Configuration Register 2 ..... 2640
29.4.15 ECMnIRCFG0 — ECM n Internal Reset Configuration Register 0 ..... 2641
29.4.16 ECMnIRCFG1 — ECM n Internal Reset Configuration Register 1 ..... 2642
29.4.17 ECMnIRCFG2 — ECM n Internal Reset Configuration Register 2 ..... 2643
29.4.18 ECMnEMK0 — ECM n Error Mask Register 0 ..... 2644
29.4.19 ECMnEMK1 — ECM n Error Mask Register 1 ..... 2645
29.4.20 ECMnEMK2 — ECM n Error Mask Register 2 ..... 2646
29.4.21 ECMnESSTC0 — ECM n Error Source Status Clear Trigger Register 0 ..... 2647
29.4.22 ECMnESSTC1 — ECM n Error Source Status Clear Trigger Register 1 ..... 2648
29.4.23 ECMnESSTC2 — ECM n Error Source Status Clear Trigger Register 2 ..... 2649
29.4.24 ECMnPCMD1 — ECM n Protection Command Register ..... 2650
29.4.25 ECMnPS — ECM n Protection Status Register ..... 2651
29.4.26 ECMnPE0 — ECM n Pseudo Error Trigger Register 0 ..... 2652
29.4.27 ECMnPE1 — ECM n Pseudo Error Trigger Register 1 ..... 2653
29.4.28 ECMnPE2 — ECM n Pseudo Error Trigger Register 2 ..... 2654
29.4.29 ECMnDTMCTL — ECM n Delay Timer Control Register ..... 2655
29.4.30 ECMnDTMR — ECM n Delay Timer Register ..... 2656
29.4.31 ECMnDTMCMP — ECM n Delay Timer Compare Register ..... 2657
29.4.32 ECMnDTMCFG0 — ECM n Delay Timer Configuration Register 0 ..... 2658
29.4.33 ECMnDTMCFG1 — ECM n Delay Timer Configuration Register 1 ..... 2659
29.4.34 ECMnDTMCFG2 — ECM n Delay Timer Configuration Register 2 ..... 2660
29.4.35 ECMnDTMCFG3 — ECM n Delay Timer Configuration Register 3 ..... 2661

29.4.36 ECMnDTMCFG4 — ECM n Delay Timer Configuration Register 4 ..... 2662
29.4.37 ECMnDTMCFG5 — ECM n Delay Timer Configuration Register 5 ..... 2663
29.4.38 ECMnEOCCFG — ECM n Error Output Clear Invalidation Configuration Register ..... 2664
29.4.39 ECMnPEM — ECM n Pseudo error mask register ..... 2665
29.5 Difference among P1M-C, P1H-C and P1H-CE ..... 2666

Section 30 Data CRC Function B (DCRB) ..... 2667
30.1 Feature ..... 2667
30.1.1 Number of Units and Channels ..... 2667
30.1.2 Register Base Address ..... 2667
30.1.3 Clock Supply ..... 2668
30.1.4 Interrupt and DMA/DTS Requests ..... 2668
30.1.5 External Input and Output Pins ..... 2668
30.2 Overview ..... 2669
30.2.1 Functional Overview ..... 2669
30.2.2 Block Diagram ..... 2671
30.3 Registers ..... 2672
30.3.1 List of Registers ..... 2672
30.3.2 DCRBnCIN — DCRB input register ..... 2673
30.3.3 DCRBnCOUT — DCRB data register ..... 2675
30.3.4 DCRBnCTL — DCRB control register ..... 2676
30.4 Operation ..... 2677
30.5 Difference among P1M-C, P1H-C and P1H-CE ..... 2678

Section 31 On-Chip Debugging Unit (OCD) ..... 2679
31.1 Debug Function ..... 2679
31.2 Calibration Function (P1H-CE only) ..... 2681
31.3 Trace Control Function ..... 2681
31.4 Peripheral Break Control ..... 2681
31.5 Usage Notes for On-chip Debugging ..... 2682

Section 32 Flash Memory ..... 2683
32.1 Features ..... 2683
32.1.1 Clock Supply ..... 2683
32.2 Block Diagram ..... 2684
32.3 Flash Size ..... 2684
32.4 Memory Configuration ..... 2685
32.5 Registers ..... 2687
32.5.1 Register Base Address ..... 2687
32.5.2 List of Registers ..... 2687
32.5.3 Register Reset Condition ..... 2688
32.6 Operating Modes Associated with Flash Memory ..... 2689
32.7 Functional Overview ..... 2690

32.8 Communications Mode ..... 2695
32.8.1 One-wire UART as an Asynchronous Flash Programming Interface ..... 2695
32.8.2 Two-wire UART as an Asynchronous Flash Programming Interface ..... 2695
32.8.3 CSI as a Synchronous Flash Programming Interface ..... 2695
32.8.4 Selecting the Communications System ..... 2696
32.9 Programming with Serial Programmer ..... 2697
32.9.1 Programming Environment ..... 2697
32.10 Programming with Self-programming ..... 2698
32.10.1 Overview ..... 2698
32.10.2 BGO Function ..... 2698
32.10.3 Enabling of Self-Programming ..... 2699
32.10.4 Procedure for Writing to a Write-Protected Register ..... 2703
32.11 Flash Memory Read ..... 2704
32.11.1 Code Flash Memory Read ..... 2704
32.11.2 Data Flash Memory Read ..... 2705
32.11.3 Registers Related to Write and Erase Protect of Flash Memory ..... 2706
32.11.4 Register Related to Reset Vector ..... 2708
32.11.5 Registers Related to Product Information ..... 2709
32.12 Option Bytes ..... 2711
32.12.1 OPBT0 — Option Byte 0 ..... 2711
32.12.2 OPBT1 — Option Byte 1 ..... 2713
32.12.3 OPBT2 — Option Byte 2 ..... 2714
32.12.4 OPBT13 — Option Byte 13 ..... 2715
32.12.5 OPBT14 — Option Byte 14 ..... 2716
32.12.6 OPBT15 — Option Byte 15 ..... 2718
32.13 Erase Counter ..... 2719
32.14 ECC Test Area ..... 2720
32.15 Usage Notes ..... 2721
32.16 Reference documents ..... 2723
32.17 Difference among P1M-C, P1H-C and P1H-CE ..... 2723

Section 33 RAM Modules ..... 2724

33.1 Features ..... 2724
33.1.1 List of On-chip RAM ..... 2724
33.2 Overview ..... 2725
33.2.1 Function overview ..... 2725
33.3 Emulation RAM (P1H-CE only) ..... 2726
33.3.1 Code Flash Emulation Function Using the Emulation RAM ..... 2726
33.4 Usage Notes ..... 2727
33.5 Difference among P1M-C, P1H-C and P1H-CE ..... 2727

Section 34 Boundary Scan ..... 2728

34.1 Functional Overview ..... 2728

34.1.1 Features ..... 2728
34.1.2 Block diagram ..... 2729
34.1.3 External input/output ports ..... 2730
34.1.4 Boundary Scan Mode ..... 2730
34.2 TAP Controller ..... 2731
34.2.1 Target pins ..... 2731
34.2.2 Instructions ..... 2731
34.2.3 Scan registers ..... 2733
34.2.4 Status transitions ..... 2734
34.3 Difference among P1M-C, P1H-C and P1H-CE ..... 2735

Section 35 Electrical Specifications ..... 2736

35.1 Overview ..... 2736
35.2 Absolute Maximum Ratings ..... 2736
35.3 General Characteristics ..... 2738
35.3.1 Supply Voltage Characteristics ..... 2738
35.3.2 Main Oscillator Characteristics ..... 2739
35.3.3 PLL0 Characteristics ..... 2740
35.3.4 Internal Oscillator Characteristics ..... 2741
35.3.5 Operational Condition ..... 2741
35.3.6 IO Capacitances ..... 2742
35.4 DC Characteristics ..... 2743
35.4.1 Input Leakage Current ..... 2743
35.4.2 Digital IO Pins Input and Output Characteristics ..... 2744
35.4.3 Analog Input Pins (ADCnIm) Characteristics ..... 2748
35.4.4 Main Oscillator Input Pin (X1) Characteristics ..... 2748
35.4.5 Aurora Interface Clock Characteristics ..... 2749
35.4.6 Injected Current Characteristics ..... 2749
35.5 Supply Current Characteristics ..... 2750
35.5.1 General Definition ..... 2750
35.5.2 Power Supply Currents (P1M-C) ..... 2751
35.5.3 Power Supply Currents (P1H-C (4MB)) ..... 2752
35.5.4 Power Supply Currents (P1H-C (8MB)) ..... 2753
35.5.5 Power Supply Currents (P1H-CE) ..... 2754
35.6 AC Characteristics ..... 2755
35.6.1 AC Test Conditions ..... 2755
35.6.2 Power Up/Down Timing ..... 2757
35.6.3 RESETZ Timing ..... 2759
35.6.4 Interrupt Timing ..... 2760
35.6.5 Mode Timing ..... 2761
35.6.6 ADCnTRG Timing ..... 2762
35.6.7 Clock Output Timing ..... 2763
35.6.8 High-Speed USRT Timing ..... 2763
35.6.9 CSIH Timing ..... 2765
35.6.10 FlexRay Timing ..... 2777

35.6.11 RLIN3 Timing ..... 2777
35.6.12 Ethernet Timing ..... 2778
35.6.13 GTM Timing ..... 2781
35.6.14 MCAN Timing ..... 2781
35.6.15 External Memory Controller (MEMC0) Timing ..... 2782
35.6.16 Emergency Shut-Off (ESO) Timing ..... 2786
35.6.17 TRSTZ Timing ..... 2787
35.6.18 Nexus Interface Timing ..... 2788
35.6.19 LPD (4pin) Interface Timing ..... 2789
35.6.20 AUDR Interface Timing ..... 2790
35.6.21 Aurora Interface Timing ..... 2792
35.6.22 Debug Resource (Aurora, ERAM) Specific Reset Timing ..... 2794
35.6.23 Debug Event Interface Timing ..... 2796
35.6.24 Flash Programming Characteristics ..... 2797
35.7 Analog Functions Characteristics ..... 2799
35.7.1 POC Characteristics ..... 2799
35.7.2 Core Voltage Monitor (CVM) Characteristics ..... 2800
35.7.3 A/D Converter Characteristics ..... 2801
35.7.4 Temperature Sensor Characteristics ..... 2803
35.7.5 Flash Characteristics ..... 2804
35.8 Thermal Characteristics ..... 2808
35.8.1 Junction-to-Board Resistance (Ψjb) ..... 2808

Section 36 Package ..... 2810
36.1 Package Line Up ..... 2810
36.2 Package Outline ..... 2811
36.2.1 LQFP Package Drawing ..... 2811
36.2.2 FBGA (156pin) Package Drawing ..... 2812
36.2.3 FBGA (292pin) Package Drawing ..... 2813
36.2.4 FBGA (404pin) Package Drawing ..... 2814
36.3 Differences among products ..... 2815

Section 37 CAN Controller (MCAN) ..... 2816
37.1 Features ..... 2816
37.1.1 Number of Units and Channels ..... 2817
37.1.2 Register Base Address ..... 2817
37.1.3 Clock Supply ..... 2818
37.1.4 RAM areas for MCAN ..... 2818
37.1.5 Interrupt Requests ..... 2819
37.1.6 External Input / Output Pins ..... 2819
37.2 Overview ..... 2820
37.2.1 Functional overview ..... 2820
37.2.2 Block Diagram ..... 2822
37.3 Registers ..... 2823

37.3.1 List of Registers ..... 2823
37.4 Operation ..... 2824
37.4.1 Procedure of Module Standby and Limited Reset ..... 2824
37.5 MCAN ..... 2826
37.5.1 Overview ..... 2826
37.5.2 Programmer’s Model ..... 2829
37.5.3 Functional Description ..... 2894
37.6 M_TTCAN ..... 2915
37.6.1 Overview ..... 2915
37.6.2 Programmer’s Model ..... 2920
37.6.3 Functional Description ..... 3009
37.6.4 TTCAN Operation ..... 3032
37.7 Usage Notes ..... 3056
37.7.1 Port sharing between M_CAN IP and RLIN ..... 3056
37.8 CAUTION ..... 3057
37.9 Difference among P1M-C, P1H-C and P1H-CE ..... 3064

# Section 1 Overview

RH850/P1x-C is a product series of Renesas Electronics' single-chip microcomputer RH850 family. This section describes the overview of RH850/P1x-C.

## 1.1 Outline

RH850/P1x-C is a 32-bit single-chip microcomputer with multiple CPUs, Code Flash, Data Flash, RAM modules, DMA controllers, many communication interfaces that are used in the automotive applications, A/D converters, timer units, etc. This chip is oriented to the automotive applications which comply with functional safety standard (ISO26262).

The main features are as below.

(1) RH850 multi-core CPU
This chip equips one or two RH850 G3M unit(s) as main CPU(s) depending on the product type, and each CPU unit has Lock-Step Dual Core feature for the functional safety. In addition to that, the chip equips one RH850 G3K unit as a sub CPU which controls security functionality (depending on product type).
The architecture of these CPUs realizes the compact footprint of the programs by 2-byte basic instructions and high-level language compiler oriented instruction sets. These CPUs have very quick interrupt response time so that they can support hard real-time applications.

(2) On-Chip Code Flash and Data Flash
This chip has high-speed Code Flash from which the CPU can fetch the instructions and the constant data. Its capacity is up to 8MB for the Production Device and 10MB for the Emulation Device. This Code Flash can be reprogrammed in the situation that the chip is mounted in the application systems.
This chip also has Data Flash with EEPROM emulation capability. Its capacity is up to 224KB.

(3) Rich peripheral functionality
This chip equips not only common communication interfaces such as SPI and HS-USRT but also automotive oriented ones such as MCAN, FlexRay, RLIN, SENT. In addition, it also supports Ethernet (depending on product type). As internal peripheral modules, the chip has A/D Converter, System Timer, Generic Timer Module, and dedicated Peripheral Interconnect module which connects the functionalities of these peripherals.

(4) Functional Safety support
This chip equips several dedicated functionalities including the Lock-Step Dual Core configuration for the CPU, the memory protection with ECC, the bus protection with ECC, the peripheral module protection, and voltage / clock monitors to support the functional safety standard (ISO26262) required in the automotive applications.

(5) Security support
This chip supports various security features (depending on product type). The Intelligent Cryptographic Unit – Master (ICUMC) has a dedicated secure CPU (RH850G3K) and some secure peripherals such as AES engine and Random Number Generator (RNG). The chip also realizes the HW-level domain separation between non-secure and secure domains. The internal resources such as Code- and Data-Flash can be assigned to either domain and the secure domain is protected against non-secure accesses by the HW mechanism. The chip also has the protection scheme for a malicious access from external.

## 1.2 Product List

Table 1.1 Product list and package

| Product Name | Product type | Package | Package Name | Note |
|---|---|---|---|---|
| R7F701370A | P1H-CE | BGA404 | R7F701370AEEBG | |
| R7F701370B | P1H-CE | BGA404 | R7F701370BEEBG | CAN-FD |
| R7F701371 | P1H-C (8MB) | BGA292 | R7F701371EABG | CAN-FD |
| R7F701372 | P1H-C (4MB) | BGA292 | R7F701372EABG | |
| R7F701372A | P1H-C (4MB) | BGA292 | R7F701372AEABG | CAN-FD |
| R7F701396A | P1H-C (4MB) | BGA156 | R7F701396AEABG | CAN-FD |
| R7F701373 | P1M-C | BGA292 | R7F701373EABG | |
| R7F701373A | P1M-C | BGA292 | R7F701373AEABG | CAN-FD |
| R7F701374 | P1M-C | LQFP144 | R7F701374EAFP | |
| R7F701374A | P1M-C | LQFP144 | R7F701374AEAFP | CAN-FD |
| R7F701397A | P1M-C | BGA156 | R7F701397AEABG | CAN-FD |

For better readability common Product Short Names will be used in this manual.

Wherever a more detailed distinction of the device is required either the specific Product Name (R7Fxxxx) or the nickname followed by further information in brackets (e.g. “P1M-C (BGA-292)”) will be used.

| Product | | RH850/P1x-C | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Product Short Name | | P1M-C | | | P1H-C (4MB) | | P1H-C (8MB) | P1H-CE |
| Package | | QFP-144 | BGA | | BGA | | BGA-292 | BGA-404 |
| | | | BGA-156 | BGA-292 | BGA-156 | BGA-292 | | |
| | Package Extension to Product Name | EAFP | EABG | | | | | EEBG |
| Product name | Non-CANFD | R7F701374 | — | R7F701373 | — | R7F701372 | — | R7F701370A |
| | CANFD | R7F701374A | R7F701397A | R7F701373A | R7F701396A | R7F701372A | R7F701371 | R7F701370B |

- “P1x-C” refers to all products covered in this document.
- “P1M-C” refers to all P1M-C variants regardless of package (QFP, BGA).
- “P1M-C(BGA-292)” refers to a subset of P1M-C devices, i.e. all “P1M-C” in the BGA package with 292 balls

Table 1.2 Features in each product (1/2)

| Category | Item | Sub-item | unit | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|---|
| CPU Subsystem | PE1 | CPU Core | — | RH850 G3M | RH850 G3M | RH850 G3M | RH850 G3M |
| | | LSDC | — | Yes | Yes | Yes | Yes |
| | | Clock Freq. | MHz | 120, 160, 240 | 160, 240 | 160, 240 | 120, 160, 240 |
| | | FPU | — | Yes | Yes | Yes | Yes |
| | | I-Cache | KB | 16 | 16 | 16 | 16 |
| | | MPU | ch | 16 | 16 | 16 | 16 |
| | | INTC1 | ch | 32 | 32 | 32 | 32 |
| | | Local RAM | KB | 128 | 64 | 64 | 128 |
| | PE2 | CPU Core | — | — | RH850 G3M | RH850 G3M | RH850 G3M |
| | | LSDC | — | — | Yes | Yes | Yes |
| | | Clock Freq. | MHz | — | 160, 240 | 160, 240 | 120, 160, 240 |
| | | FPU | — | — | Yes | Yes | Yes |
| | | I-Cache | KB | — | 16 | 16 | 16 |
| | | MPU | ch | — | 16 | 16 | 16 |
| | | INTC1 | ch | — | 32 | 32 | 32 |
| | | Local RAM | KB | — | 64 | 64 | 64 |
| | Global RAM | | KB | 320 | 960 | 960 | 1344 |
| | Total RAM | LRAM + GRAM | KB | 448 | 1088 | 1088 | 1536 |
| | INTC2 | | ch | 224 | 224 | 224 | 224 |
| | DMA | DMAC | ch | 16 | 16 | 16 | 16 |
| | | DTS | ch | 128 | 128 | 128 | 128 |
| Flash Memory | Code Flash | | MB | 2 | 4 | 8 | 8 |
| | Extra Code Flash (ED only) | | MB | — | — | — | 2 |
| | ERAM (Calibration RAM: ED only) | | MB | — | — | — | 2 |
| | Data Flash | NonSecure | KB | 64 | 128 | 192 | 192 |
| | | Secure | KB | 32 | 32 | 32 | 32 |
| Security | ICUMC | CPU Core | — | RH850 G3K | RH850 G3K | RH850 G3K | RH850 G3K |
| | | Clock Freq. | MHz | 60, 80 | 80 | 80 | 60, 80 |
| | BasicHardwareProtecton | | — | Yes | Yes | Yes | Yes |
| | Secure WDT | | ch | 1 | 1 | 1 | 1 |
| External Bus | | | — | No | P1H-C (4MB, BGA-292): Yes<br>P1H-C (4MB, BGA-156): No | Yes | Yes |
| A/D Converter | Module | | instance | 2 | 2 | 2 | 2 |
| | Analog input | | ch | P1M-C (QFP): 20<br>P1M-C (BGA-292): 24<br>P1M-C (BGA-156): 16 | P1H-C (4MB, BGA-292): 32<br>P1H-C (4MB, BGA-156): 16 | 40 | 40 |
| | Voltage | | V | 3.3 | 3.3 | 3.3 | 3.3 |
| Timers | System Timer | | instance | 1 | 2 | 2 | 2 |
| | WDT | | instance | 1 | 2 | 2 | 2 |
| | PIC | | — | Yes | Yes | Yes | Yes |
| | GTM | | — | Config 2 | Config 3 | Config 3 | Config 3 |

Table 1.2 Features in each product (2/2)

| Category | Item | Sub-item | unit | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|---|
| Communication Interface | MCAN | | ch | 2 | 2 | 3 | 3 |
| | M_TTCAN | | ch | 1 | 1 | 1 | 1 |
| | FlexRay | | ch | 2 | P1H-C (4MB, BGA-292): 4<br>P1H-C (4MB, BGA-156): 2 | 4 | 4 |
| | Ethernet | | port | 1 | P1H-C (4MB, BGA-292): 2<br>P1H-C (4MB, BGA-156): 1 | 2 | 2 |
| | SENT | | ch | P1M-C (QFP, BGA-292): 6<br>P1M-C (BGA-156): 4 | P1H-C (4MB, BGA-292): 8<br>P1H-C (4MB, BGA-156): 4 | 8 | 10 |
| | HS-USRT | | ch | 2 | P1H-C (4MB, BGA-292): 4<br>P1H-C (4MB, BGA-156): 2 | 4 | 4 |
| | RLIN | | ch | 2 | P1H-C (4MB, BGA-292): 4<br>P1H-C (4MB, BGA-156): 2 | 4 | 4 |
| | CSIH (SPI) | # of ch | ch | 4 | 4 | 4 | 4 |
| | | # of CS | pin | P1M-C (QFP): 28 (8/8/8/4)<br>P1M-C (BGA-292): 30 (8/8/8/6)<br>P1M-C (BGA-156): 24 (8/8/6/2) | P1H-C (4MB, BGA-292): 32 (8/8/8/8)<br>P1H-C (4MB, BGA-156): 24 (8/8/6/2) | 32 (8/8/8/8) | 32 (8/8/8/8) |
| Safety | ECM | | — | Yes | Yes | Yes | Yes |
| | Data CRC | | ch | 4 | 8 | 8 | 8 |
| Internal Monitors | Clock | | — | Yes | Yes | Yes | Yes |
| | Core Voltage | | — | Yes | Yes | Yes | Yes |
| | Temperature | | — | Yes | Yes | Yes | Yes |
| Debug | Debug Control | | — | JTAG, LPD | JTAG, LPD | JTAG, LPD | JTAG, LPD |
| | Branch Trace | | — | — | — | — | Yes |
| | Data Trace | | — | — | — | — | Yes |
| | Trace I/F | | — | — | — | — | Aurora 2 lanes Max 6.25 Gbps (3.125 Gbps/ lane) |
| | Bypass I/F | | — | — | — | — | AUD Max 200 Mbps |
| Power Supply | Core | | V | P1M-C (QFP, BGA-292): 1.25 (DPS), eVR<br>P1M-C (BGA-156): 1.25 (DPS) | 1.25 | 1.25 | 1.25 |
| | I/O | | V | 3.3 | 3.3 | 3.3 | 3.3 |
| | ADC | | V | 3.3 | 3.3 | 3.3 | 3.3 |

## 1.3 Block Diagram

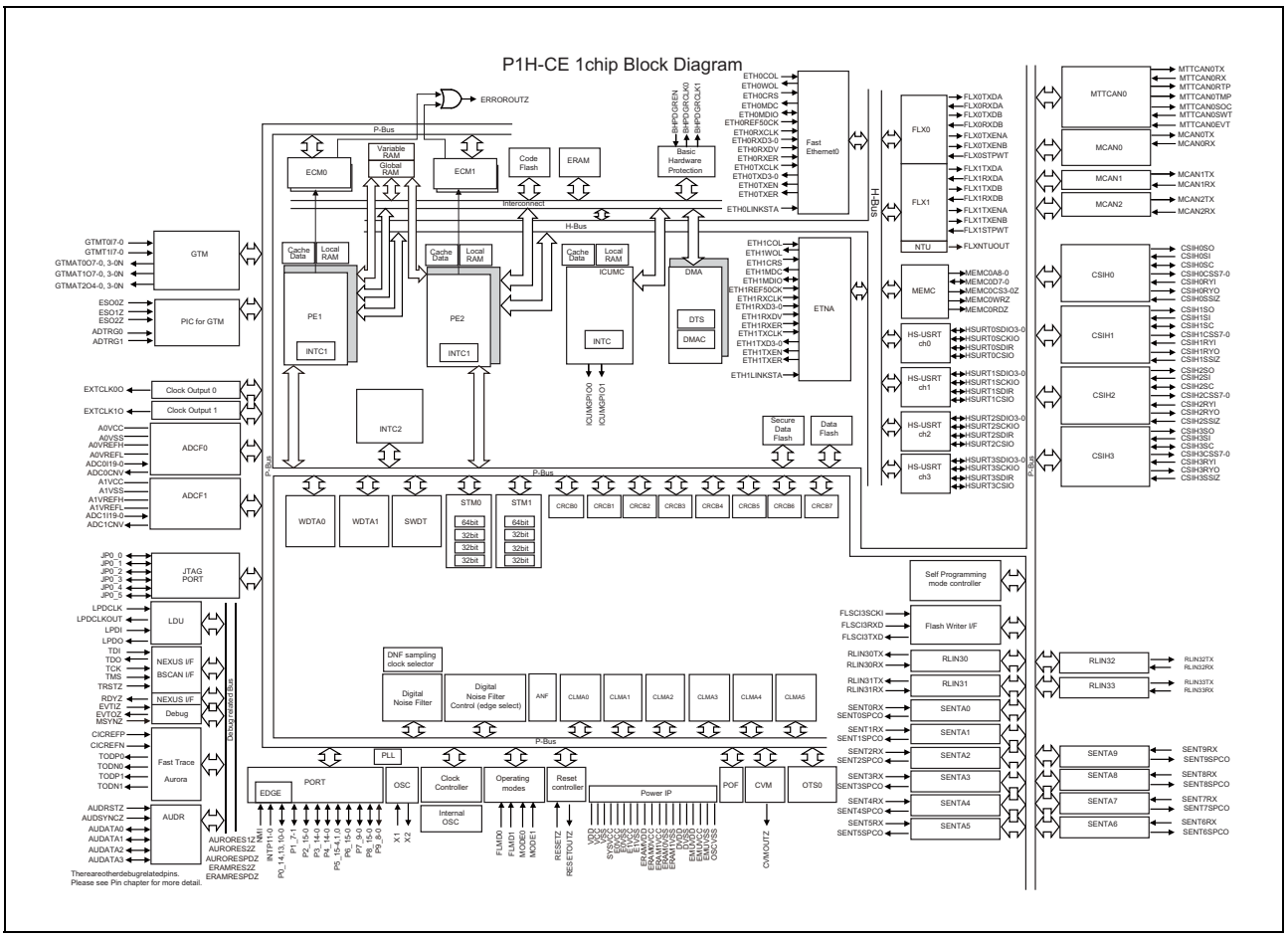



**Figure 1.1 RH850/P1x-C chip block diagram (P1H-CE)**

## 1.4 Features

(1/12)

| Item | Features |
|---|---|
| Pin and Port Functions | • Most of the digital pins have selectable functionalities from up to nine dedicated functions (up to four alternative functions with each In and Out and a special function) and a general purpose I/O port.<br>• Selectable drivability for output pins<br>• Selectable pull-up/pull-down/off for input pins<br>• General purpose I/O port can be accessed per-pin basis by multi-CPU without mutual exclusion |
| CPU Subsystem | • CPU<br>– High performance 32-bit architecture for embedded controllers<br>– 32-bit internal data busses<br>– Thirty-two 32-bit general purpose registers<br>– RISC type instructions set<br>• Load / Store instructions with long / short format<br>• Three operand instructions<br>• Based on C language<br>– Operating modes: User mode / Supervisor mode<br>– Address space: 4GB linear address space for data and instruction<br>• Floating Point Unit (FPU)<br>– Supports single-precision (32-bit) and double-precision (64-bit)<br>– Supports IEEE754 compliant data types and exceptions<br>– Rounding modes: To nearest / Toward 0 / Toward +∞ / Toward -∞<br>– Handling of denormalized numbers: Truncation to zero / IEEE754 compliant exception<br>• Interrupt Handling<br>– 16 interrupt priority levels can be specified for each interrupt channel<br>– Two methods of interrupt handler address calculation<br>• Direct vector method<br>• Table reference method<br>– PUSHSP and POPSP instructions for fast context switch<br>• Protection Mechanisms<br>– Memory Protection Unit (MPU) protects memory regions against illegal accesses by software.<br>– Internal Peripheral Guard (IPG) protects registers of internal peripherals against illegal accesses by software.<br>– PE Guard (PEG) protects Local RAM and internal peripheral registers against illegal accesses by external bus masters.<br>– Global RAM (GRAM) Guard protects GRAM regions against illegal accesses by all bus masters.<br>• Instruction Cache<br>– Each CPU has its own instruction cache for the Code Flash.<br>• Local RAM<br>– Each CPU has its own Local RAM.<br>• Global RAM<br>– Each CPU can access Global RAM.<br>• Interprocessor Communication and Mutual Exclusion Mechanisms<br>– Interprocessor interrupt registers (IPIR)<br>– Semaphore registers<br>– CAXI (compare and exchange for interlock), LDL.W (load linked word) and STC.W (store conditional word) instructions |

(2/12)

| Item | Features |
|---|---|
| Operating Modes | • Operating modes<br>– Normal Operating Mode<br>– Serial Flash Programming Mode<br>– Boundary SCAN Mode<br>• Mode Setting<br>– Set by the mode selecting pins: FLMD0, FLMD1, MODE0, MODE1 (FLMD1, MODE0, MODE1 can be used as GPIO as well.)<br>– Latched at Power-On Reset or System Reset 1 |
| External Memory Controller | • Supports external memory access (Devices with conventional SRAM interface directly connectable)<br>• Four Chip Select pins (CS0, CS1, CS2, CS3)<br>Up to 512 bytes linear address space for each CS<br>• 8-bit data bus<br>• Programmable data setup wait cycle independently for<br>– Each CS<br>– Read / Write<br>• Programmable write data hold wait cycle (deassertion edge of write strobe)<br>• Programmable Idle cycle independently for after read and after write bus cycle |
| Interrupt Functions | • Three types of interrupt<br>– FENMI: FE-level non maskable interrupt<br>– FEINT: FE-level maskable interrupt<br>– EIINT: EI-level maskable interrupt<br>• 16 priority levels for each interrupt independently<br>• Maskable for each interrupt independently<br>• Interrupt Controller<br>– Primary interrupt controller (INTC1)<br>• Each CPU (PE1 and PE2) has its own INTC1.<br>• Direct handling up to 2 FE-level interrupts and up to 32 EI-level interrupts<br>– Secondary interrupt controller (INTC2)<br>• Shared by both CPUs<br>• Up to 224 interrupts notified to CPU via INTC1<br>• Each interrupt can be independently mapped to only one CPU or both CPUs in parallel. |
| Direct Memory Access | • Two types of DMA engine: DMAC and DTS<br>– DMAC<br>• Transfer information is stored in the DMAC control registers.<br>• Up to 16 channels (1 register set for each channel)<br>• Up to 128 DMA trigger source<br>– DTS<br>• Transfer information is stored in the dedicated SRAM (DTSRAM).<br>• Up to 128 channels can be active at the same time.<br>• Up to 128 DTS requests<br>• Address space: 4GB<br>• Transfer mode: Single transfer, Block transfer 1, Block transfer 2<br>• Transfer data size: 1/2/4/8/16 bytes<br>• Interrupt: Transfer Completion Interrupt, Transfer Count Match Interrupt<br>• Reload function support<br>• Chain function support |
| Power Supply | • Dual Power Supply (DPS)<br>• Internal POC (Power On Clear) for safe startup<br>• Core Voltage Monitor (CVM) for the internal supply voltage (VDD) monitoring |

(3/12)

| Item | Features |
|---|---|
| Reset Controller | • 6 reset functions<br>– Power-On Reset<br>– System Reset 1<br>– System Reset 2<br>– Application Reset 1<br>– Limited Reset<br>– Debug Reset<br>• External output pin: RESETOUTZ<br>• Automatic RAM initialization after some reset<br>• Automatic HW BIST execution after some reset |
| Clock Controller | • On-chip crystal resonant circuit (Main OSC)<br>• On-chip internal oscillator which is used as a Backup clock during the clock start up and used for safety purpose<br>• On-chip PLL to generate the high speed internal clocks from Main OSC<br>• Clock frequency<br>– Main OSC: 16/20/24MHz<br>– Internal oscillator: 16MHz<br>– Internal clocks generated by the PLL: up to 240MHz<br>• Two software configurable external clock outputs<br>• Software configurable clock dividers to enable flexible clock gear function |
| Power Down Modes | • HALT mode support in Main CPUs (PE1 and PE2) and ICUMC (ICUP)<br>• Module Standby mode support in ICUMC and some peripheral module |
| Clocked Serial Interface H | • Three-wire serial synchronous data transfer well known as SPI<br>• Full duplex operation (simultaneous transfer and receive), receive only mode, or transmit only mode<br>• Master mode and slave mode selectable<br>• Phase of clock and data selectable for each chip select<br>• Data transfer with MSB or LSB first selectable for each chip select<br>• Transfer data length selectable from 2 to 16 bits in 1-bit units for each chip select<br>• EDL (Extended Data Length) function for transferring data with more than 16 bits<br>• Maximum transmission speed<br>– in master mode: up to 20Mbit/s<br>– in slave mode: up to 20Mbit/s<br>• Bit rate selectable by BRG output (at Master mode) or by slave clock<br>• Transmit mode, Receive mode and Transmit/Receive mode selectable<br>• Buffer size is 128 words (1 word is data 32 bits + ECC 7 bits)<br>• Memory mode selectable (FIFO, dual buffer, Tx-only buffer, direct access)<br>• Built-in handshake function<br>• Error detection (data consistency check, parity, time-out, overrun)<br>• JOB enable control bit for AUTOSAR<br>• RCB (Recessive Configuration for Broadcasting) bit for Broadcasting<br>• LBM (Loop Back Mode) function for self test<br>• Four different interrupt request signals (INTCSIHnTIC, INTCSIHnTIR, INTCSIHnTIRE, INTCSIHnTIJC)<br>• IDLE State Control function<br>• Silent mode communication for extended idle time<br>• Automatically generation of chip select output signal with configurable active level<br>• Data transfer without activated chip select<br>• Transmission speed for each chip select is selectable out of four predefined baud rates (in master mode) or by clock input signal from master (in slave mode)<br>• Full DMA support for all CSIH registers |

(4/12)

| Item | Features |
|---|---|
| High-Speed Universal Synchronous Receiver / Transmitter | • Communication and framing<br>– Half-duplex synchronous serial communication<br>– Selectable : Master (transmission) and Slave (reception)<br>The status (master or slave) is shown by SDIR pin.<br>– MSB first<br>– Communication frame length<br>• Data of 16 bits w/o parity bit<br>• Data of 15 bits with even parity bit<br>• Data of 15 bits with odd parity bit<br>• Number of the communication lane<br>– Selectable from 1, 2, and 4 lanes<br>• Communication rate<br>– Tx: 10 Mbps / lane (40 Mbps in 4 lanes) [TXCLK@80MHz]<br>– Rx: 40 Mbps / lane (160 Mbps in 4 lanes)<br>• Clock divider of TXCLK for the master communication clock<br>– Selectable from 8, 10, 12, 14, 16, 24, and 32<br>• Communication buffer size<br>– Transmission: 8 frames (4 frames x2)<br>– Reception: 64frames (32frames x2)<br>• Interrupt factor<br>– TX Completion Interrupt<br>– RX of First Frame Interrupt<br>– RX of Group Frames Interrupt<br>– RX Completion Interrupt<br>– Parity Error Interrupt<br>– RX Buffer Overflow Error Interrupt<br>– Error detection<br>– Parity Error (none/odd/even)<br>– RX Buffer Overflow Error<br>• Other function<br>– Integrated DMAC for communication data transfer<br>– Protect function against register write |
| RLIN | • Compliant with LIN protocol spec versions 1.3, 2.0, 2.1, 2.2 and SAE J2602<br>• Three operating modes<br>– LIN Master<br>– LIN Slave<br>– UART (half-duplex, full-duplex)<br>• LIN Self-test mode with internal data loop back |

(5/12)

| Item | Features |
|---|---|
| MCAN | • Conforming to the ISO 11898-1<br>• Data transfer rate is up to 1Mbps, individually for each CAN channel. For CAN FD, up to 8Mbps<br>• Selectable ID type<br>– 11-bit Standard ID<br>– 11-bit Standard ID + 18-bit Extended ID<br>• Message Buffer<br>– Up to 64 dedicated Receive Buffers<br>– Up to 32 dedicated Transmit Buffers<br>• Supports all AUTOSAR requirements<br>– Transmit Abort Interrupt<br>– Non-waiting processing functionality<br>– Including more than 2 TX Buffers prioritization<br>• Supports several measures for self-testing:external and internal loop back<br>• Improved RX System<br>– Scalable RX FIFO structures, with up to 64 CAN Buffers per FIFO<br>– RX timestamp<br>– RX FIFO Timeout Interrupt<br>– FIFO filling level Interrupt<br>• Improved TX System<br>– FIFO filling level supervision (interrupt)<br>– Support for transmit cancellation to avoid “inner priority inversion”<br>– Combined Message Buffer & TX FIFO and TX Queue Concept<br>– Dedicated TX message buffers for high-priority messages<br>– ID prioritization between TX buffers, TX Queue buffers and oldest TX FIFO element<br>– Transmit pause to separate two consecutive TX messages<br>• FIFOs<br>– Two configurable Receive FIFOs<br>– Configurable Transmit FIFO<br>– Configurable Transmit Queue<br>– Configurable Transmit Event FIFO<br>• Enhanced reception filtering<br>– Support of 11bit and 29bit CAN identifier, each filter element is configurable for acceptance/rejection<br>– Programmable 29 bit CAN identifier acceptance filter mask for each entry<br>– Each acceptance filter element targets FIFO 0 or 1 or a dedicated RX Buffer<br>– Every FIFO or RX Buffer filter element can be used as a from-to range filter, as a filter for one or two dedicated IDs or as a classic bit mask filter<br>– Each filter element can be enabled/disabled individually<br>• Supports TT_CAN level 2 according to ISO11898-4<br>• Supports Pretended Networking of AUTOSAR<br>• Supports Time Stamp function |

(6/12)

| Item | Features |
|---|---|
| FlexRay | • Conforming to the FlexRay protocol specification v2.1<br>• Data transfer rate: up to 10 Mbit/s on each channel<br>• Data link layer clock frequency: 80MHz<br>• FlexRay channels: 2 (channel A and B)<br>• Message buffer<br>– Up to 128 message buffers configurable<br>– Configuration of message buffers with different payload lengths possible<br>– Each message buffer can be configured as receive buffer, as transmit buffer or as part of the receive FIFO<br>– Filtering for slot counter, cycle counter, and channel<br>• Message RAM<br>– 8 Kbyte of Message RAM for storage of e.g.<br>• 128 message buffers with max. 48 byte data section<br>• up to 30 message buffers with 254 byte data section<br>• FIFO: One configurable receive FIFO<br>• Message buffer access<br>– by Host CPU via Input and Output Buffer<br>• Input Buffer: Holds message to be transferred to the Message RAM<br>• Output Buffer: Holds message read from the Message RAM<br>– by data transfer function<br>• Input transfer: Message buffer content is transferred from LRAM/GRAM to Message RAM on CPU request<br>• Output transfer: Message buffer content is transferred from Message RAM to LRAM/GRAM automatically<br>• Network management: Automatic HW support<br>• Interrupts: Maskable module interrupts<br>• Timer<br>– 2 absolute timers<br>– 1 relative timer<br>• NTU generation<br>– Generation of FlexRay macrotick<br>– Generation of FlexRay cycle |

(7/12)

| Item | Features |
|---|---|
| Ethernet | • MAC (Media Access Controller)<br>– Conforming to the IEEE 802.3 MAC layer standard<br>– 100Mbps or 10Mbps data transfer.<br>– full or half-duplex mode.<br>– PHY interface: MII (Media Independent Interface), RMII (Reduce Media Independent Interface)<br>• MAC port number: 2 ports<br>• DMA transfer function<br>– Each channel implements the H-Bus master function and transfers the received/transmitted data from/to the RAM memory area.<br>– Supports the 8 times of the 32-bit burst transfer.<br>– Descriptor management system used by EDMAC (Data transfer without CPU intervention).<br>– Single-frame/single-descriptor and single-frame/multi-descriptor oparation.<br>– Transmission/reception status indication in descriptor.<br>• Address filter function<br>– Performs address filtering based on the destination address.<br>– Up to 32 different addresses can be configured individually.<br>– Supports the storm filter for broadcast frames.<br>– Supports the mode enable/disable function for broadcast frames by using CAM entry table.<br>• Others<br>– Magic packet detection and Wake-On-LAN signal output.<br>– PAUSE control frame format.<br>– CRC calculation for RX/TX frames.<br>– Interrupts for configurable error and status conditions.<br>– Internal loopback function (external loopback function can be realized with PHY interface).<br>– MAC management counters |

(8/12)

| Item | Features |
|---|---|
| SENT | • Support of SENT standard protocol<br>– SAE J2716 JAN2010<br>• Supported features:<br>– SENT master function (= sensor data reception)<br>– Decoding the sensor data<br>– CRC for received data<br>– Falling-edge detection and timing measurement capabilities to receive the encoded signal<br>– Triple speed SENT<br>– Clock period rates in range of 1us to 90us<br>– Variable data transfer rates in range of<br>• 24.5 to 65.8kbps (based on 6 nibble data at 3us clock rate)<br>• 73.5 to 197.4kbps (based on 6 nibble data at 1us clock rate)<br>– Unidirectional communication between sensor and MCU by standard<br>– Bidirectional communication between sensor and MCU supported by SPC enhancement of the standard<br>– Single edge data transmission, coded by the temporal distance of two consecutive detected falling edges on the data line<br>– Transmission of frames with up to 6 data nibbles and additional status/communication nibble<br>– CRC protected data transmission<br>– CRC data is readable by SW<br>– Calibration phase at each data frame<br>– Multi-slave bus topology<br>– One optional pause pulse<br>• Each SENT macro consists of one SENT channel. If more than one SENT channel is required several SENT macros can be implemented. See **Table 22.2** for the number of implemented SENT channels.<br>• Each time stamp counter of the macros can run independently, or in order to synchronize the timestamp across multiple channels, one instance can be set as master and reset the configured consecutive time stamp counters. Depending on the total number of instances, more than one master-slave(s) pairs can be configured.<br>• Each SENT macro supports below requirements in addition to SAE J2716 specification of JAN2010:<br>– R1: 32-Bit-Register for serial sensor data (24Bit+CRC6+READ-Bit)<br>– R2: Includes comparator for sensor data evaluation<br>– R3: CRC check of received sensor data implemented but CRC code transparent<br>– R4: 32-Bit counter for time stamp (resolution: 1us)<br>– R5: clock ticks down to 1us<br>– R6: SPC (Short PWM Code) extension<br>• Enables bidirectional communication channel<br>• Master can pull down the signal to initiate SENT message transmission. |
| Watchdog Timer | • One channel for one CPU<br>• Fixed activation code and variable activation code (VAC) selectable<br>• Two counter start modes available<br>– Automatic (default) start mode<br>– Software trigger start mode<br>• Generate an error signal to ECM on error detection<br>• Interrupt request generation at 75% of the counter overflow value<br>• Window function for refresh |
| System Timer | • One 64-bit counter, which can be used as a long period timestamp<br>• Three 32-bit counters<br>• Four compare registers for each counter |
| Generic Timer Module | GTM is a modular timer unit used to support chassis control applications by unloading the CPU from a high interrupt load. Most of the task of GTM can run independently from the CPU. Two different GTM configurations are implemented depending on product type. |

(9/12)

| Item | Features |
|---|---|
| Peripheral Interconnect | Peripheral interconnect (PIC) connects some peripherals with each other in order to achieve enhanced functionality of a stand-alone function.<br>• ADCF trigger select function<br>• Signal routing function for:<br>– ADCF conversion interrupt routed to GTM input<br>– GTM output monitor for PWM diagnostic<br>– Hi-Z control function over external pin for GTM output<br>– Baud rate measurement for an UART (RLIN3) |
| A/D Converter | • A/D conversion based on successive approximation (SAR-ADC) method<br>• 12-bit resolution<br>• Minimum conversion time: 1 us.<br>• Virtual channel concept<br>• Scan groups support<br>• Two scan modes: Multi-cycle scan mode and Continuous scan mode<br>• Asynchronous / synchronous suspend and resume function<br>• Interrupts and DMA transfers are supported.<br>• A/D converted value adding function<br>• Analog conversion voltage range can be set.<br>• Abundant safety functions are provided. |
| Functional Safety | • Development compliant with ISO26262 functional safety standard.<br>• Major safety mechanisms<br>– Error Correction Code (ECC) and Error Detection Code (EDC)<br>– Lockstep function of Redundant Data Processing Units (CPU, DMA) with compare units<br>– Memory Protection Unit (MPU) and Slave Guards for Processor Element (PEG), Internal Peripheral Modules (IPG), Global RAM (GRG) and Peripheral Bus (PBG)<br>– Field BIST at start-up test<br>– Error Control Module (ECM), for signaling error pin at failure detection by safety mechanisms<br>– Core Voltage Monitor (CVM)<br>– Clock Monitor (CLMA)<br>– Watch Dog Timer (WDTA)<br>– Data CRC (DCRB)<br>– Triple Modular Redundant (TMR) registers, for application independent parts to ensure robustness against transient faults caused by Single Event Effects (SEE)<br>– Safety Oriented Chip Layout |
| Error Control Module | Error Control Module (ECM) generates error signal to outside MCU, interrupts, and reset requests by the error input signals from various error sources and monitor logics.<br>• Interrupt request generation as against each error cause<br>• Internal Reset (System Reset 2) request generation as against each error cause<br>• Error signal generation as against error cause, which is connected to a dedicated digital output terminal<br>• Current error status is confirmable.<br>• Self-diagnosis function is implemented.<br>• Time-out function of interrupt processing<br>• ERROROUTZ release timer<br>Wait time is configurable by SW. After the configured time is passed, ERROROUTZ can be set to high by SW.<br>• Hi-Z control of GTM timer output terminals with PIC<br>• When ECM catches the safety-related error, GTM timer output terminals will be Hi-Z by Low level of ECM output. |

(10/12)

| Item | Features |
|---|---|
| Data CRC Function | • Supports the following four CRC generator polynomials in compliance with the AUTOSAR standard<br>– 32-bit Ethernet CRC<br>– 16-bit CCITT CRC<br>– 8-bit SAE J1850 CRC<br>– 8-bit 0x2F polynomial CRC |
| Core Voltage Monitor | • CVM monitors over and under voltage of the core voltage.<br>• Violating the operating range of the core voltage is informed by:<br>– Single output pin (CVMOUTZ).<br>– Core voltage over and under-voltage flags<br>– CVM Reset<br>• With the exception of diagnostic function, the CVM function is independent from internal or external RESET, except for Power-On Reset.<br>• Diagnostic function:<br>– CVM function is testable.<br>– Over and under-voltage error can be generated without influencing core voltage itself.<br>– CVM error test is done by changing reference voltage.<br>– For diagnostic the signal path to the CVMOUTZ pin can be masked.<br>– CVMOUTZ pin provides read back function to check pin level. |
| Clock Monitor | • Up to six clock monitors depending on the device configuration.<br>• Detects clock disturbance that results in lower or higher frequency than target frequency, and sends an error notification to the ECM and INTC of ICUMC<br>• Supports self-diagnosis function |
| Temperature Sensor | • Out of range detection of temperature for security<br>• Operating modes<br>– Single measurement mode<br>– Continuous measurement mode<br>• Interrupt generation<br>– Temperature Measurement End Interrupt (OTI)<br>– Temperature Rise/Drop Interrupt (OTULI)<br>– Temperature Alarm Error (OTABE) (This is routed to ICUMC and ECM)<br>– Temperature Sensor Error (OTE)<br>• Self-diagnosis function support |
| On-Chip Debug | • Debug control interface: NEXUS (JTAG) and Low-Pin-count Debug interface<br>• Single- and Multi-core debug support via single interface for debugging and calibration<br>• Synchronized multi-core debugging, triggering and controlling<br>• Trigger/Event conditions configurable for hardware break function, tracing and timing measurement<br>• Support of timing and performance measurement functions<br>• High-level security protection feature<br>– Separate authentication mechanisms for non-secure resources and secure resources<br>– Flash contents protection in debug mode |
| Emulation Device | • Extended debug interfaces<br>– JTAG / Low-Pin-count Debug interface<br>– Trace interface (Aurora): ED only<br>– Bypass interface (AUD): ED only<br>• ERAM for Code Flash calibration<br>• Extra capacity for Code Flash and GRAM<br>• Comprehensive trace information for each CPU core, DMA and System bus including timestamp information<br>• Separate power domains for ERAM and Trace interface (Aurora)<br>• Emulation mode support for each Production Device (P1M-C and P1H-C) |

(11/12)

| Item | Features |
|---|---|
| FLASH Memory | • Code Flash<br>– Capacity: up to 8MB in Production Devices, up to 10MB in Emulation Device<br>– Dual banked configuration in Dual CPU devices (P1H-C and P1H-CE)<br>– Program unit: 256 bytes, Erase unit: 8KB or 32KB (depending on the block)<br>– Per-block OTP (One Time Programmable) support<br>– Per-block Write Protection support<br>– Security features<br>• Configurable region separation between non-secure and secure areas<br>• Password protection in the debug mode<br>• Data Flash<br>– Capacity: up to 192KB for non-secure region, 32KB for secure region*1<br>– Program unit: 4 bytes, Erase unit: 64 bytes<br>• ECC support for error detection and correction<br>• Programming method<br>– Serial programming mode from an external flash writer<br>– Self programming mode from the internal CPU<br>• BGO (BackGround Operation) support<br>– Code Flash read is possible during Data Flash P/E<br>– While the flash memory is erased / programmed / read in the bank, a flash memory in another bank, can be erased / programmed / read.<br>• Extra areas to store the system configuration parameters<br>Note 1. Can be used as normal data flash in case of non-secure applications. |
| RAM Modules | • Main CPU (PE1 and PE2)<br>– Local RAM (LRAM)<br>– Instruction Cache (Tag and Data)<br>• ICUMC<br>– Local RAM (LRAM)<br>– Instruction Cache (Tag and Data)<br>• Global RAM (GRAM)<br>• Emulation RAM (ERAM) (P1H-CE only)<br>• FCU RAM<br>• Peripheral modules<br>– DTS RAM<br>– CSIH RAM<br>– MCAN RAM<br>– FlexRay RAM<br>– Ethernet RAM<br>– GTM RAM<br>• Error detection and correction for functional safety<br>• Automatic RAM initialization after reset for LRAM, GRAM and peripheral RAMs (exceptions: the head 1KB of LRAM in PE1, FCU RAM, ERAM and Ethernet RAM) |
| Boundary Scan | • Conforming to the IEEE 1149.1<br>• Support instructions: BYPASS, EXTEST, SAMPLE/PELOAD, IDCODE |

(12/12)

| Item | Features |
|---|---|
| Basic Security | • HW level domain separation between secure and non-secure domains<br>• Flash protection<br>• Mode entry protection<br>• Debugger authentication<br>• Device degradation controlled by Flash configuration options<br>• Secure watchdog timer<br>• Cache protection<br>• RAM initialization after reset |
| ICUMC | • Dedicated CPU (RH850 G3K) for secure applications (depending on product type)<br>• Secure interrupts handling<br>• Dedicated secure peripherals<br>– AES Engine<br>– Random Number Generator (RNG)<br>– Timers<br>• Watchdog Timer<br>• Additional authentication for tool connection such as a debugger |

Table 1.3 Abbreviation table

| Full name | Abbreviation |
|---|---|
| External Memory Controller | MEMC |
| Interrupt Controller | INTC |
| Direct Memory Access Controller | DMAC |
| Data Transfer System | DTS |
| Clocked Serial Interface H | CSIH |
| High-Speed Universal Synchronous Receiver / Transmitter | HS-USRT |
| FlexRay | FLX |
| Ethernet Controller | ETNA |
| Window Watchdog Timer | WDTA |
| System Timer | STM |
| Generic Timer Module | GTM |
| Peripheral Interconnect | PIC |
| AD Converter | ADCF |
| Error Control Module | ECM |
| Data CRC Function | DCRB |
| Core Voltage Monitor | CVM |
| Clock Monitor | CLMA |
| Temperature Sensor | OTS |
| On-Chip Debug | OCD |
| Emulation Device | ED |
| Intelligent Cryptographic Unit – Master | ICUMC |

## 1.5 Difference among P1M-C, P1H-C and P1H-CE

See **Table 1.2**.

# Section 2 Pin Functions

This section describes the pin and port functions.

**Section 2.1**, **2.2** describes the pin connection and respective pins.

**Section 2.3** to **2.6** describes the general port functions.

**Section 2.7** describe Noise filter & Edge detection.

## 2.1 Pin Connection Diagrams

### 2.1.1 P1M-C (QFP-144) (TOP View)

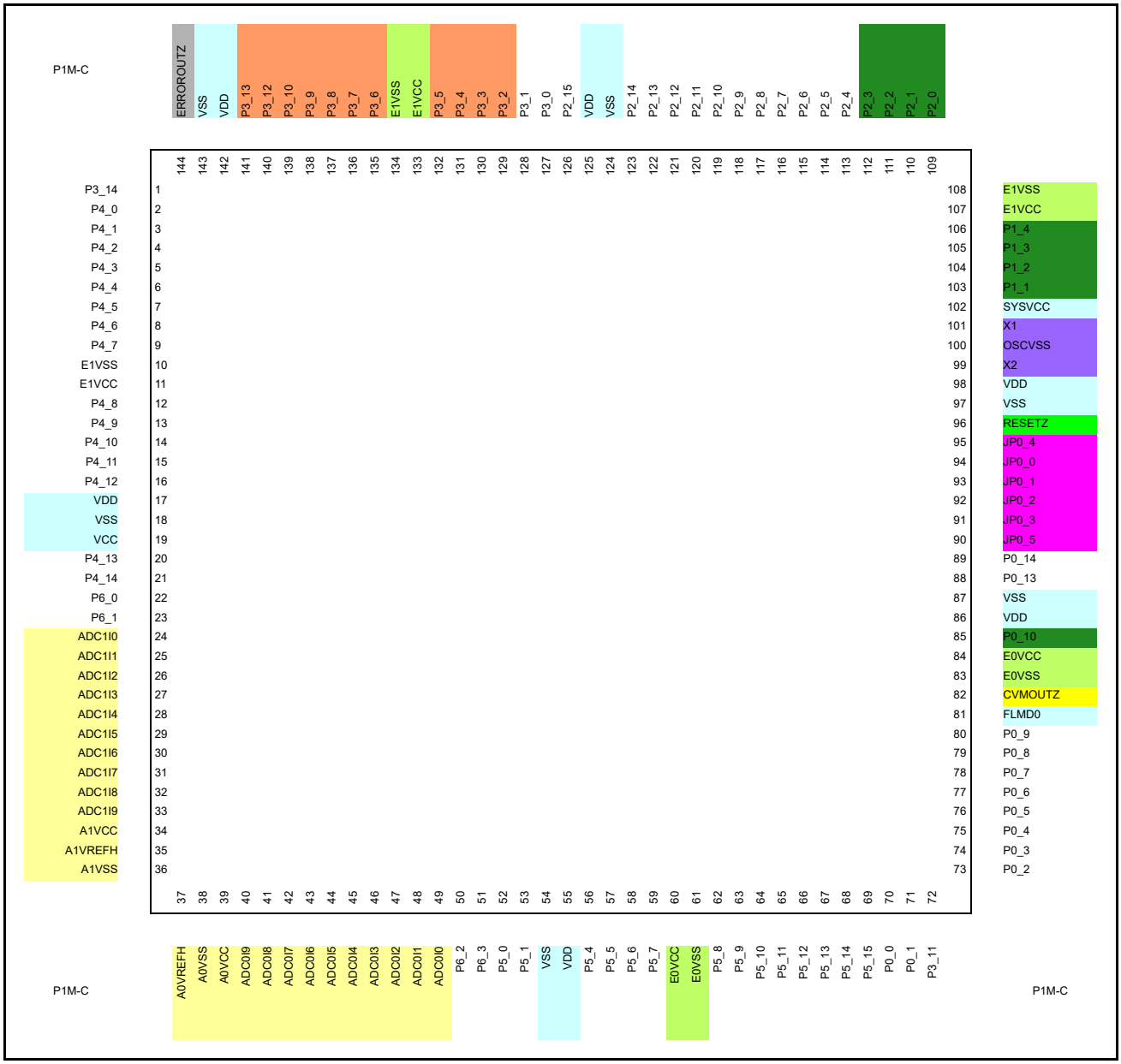
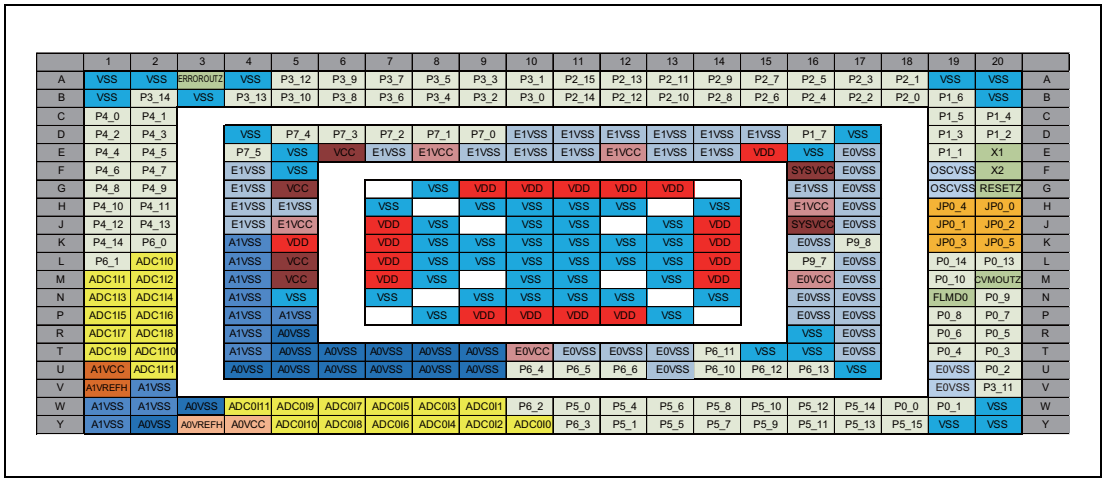



**Figure 2.1 Pin Connection Diagram of P1M-C (QFP-144)**

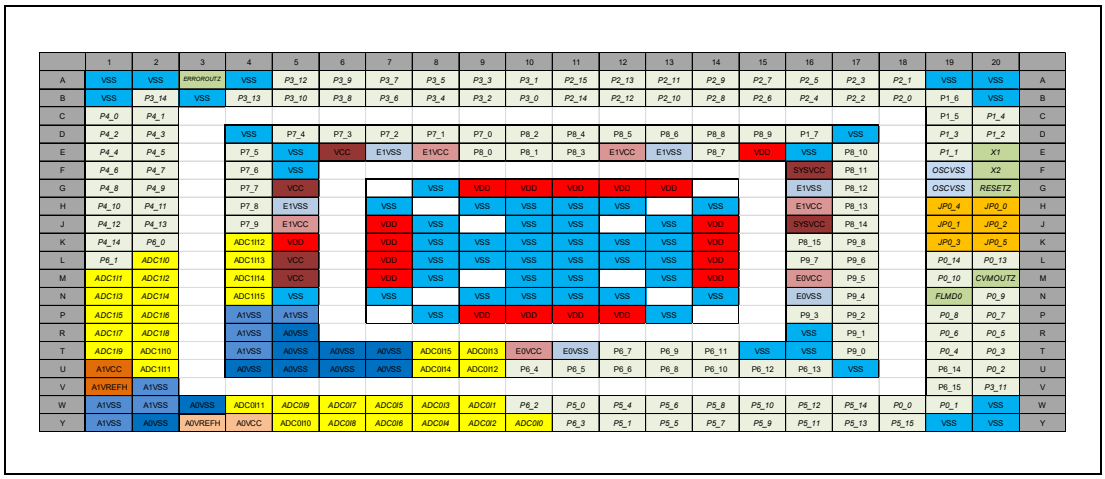
### 2.1.2 P1M-C (BGA-292) (TOP View)



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | VSS | VSS | ERROROUTZ | VSS | P3_12 | P3_9 | P3_7 | P3_5 | P3_3 | P3_1 | P2_15 | P2_13 | P2_11 | P2_9 | P2_7 | P2_5 | P2_3 | P2_1 | VSS | VSS | A |
| B | VSS | P3_14 | VSS | P3_13 | P3_10 | P3_8 | P3_6 | P3_4 | P3_2 | P3_0 | P2_14 | P2_12 | P2_10 | P2_8 | P2_6 | P2_4 | P2_2 | P2_0 | P1_6 | VSS | B |
| C | P4_0 | P4_1 | | | | | | | | | | | | | | | | | P1_5 | P1_4 | C |
| D | P4_2 | P4_3 | | VSS | P7_4 | P7_3 | P7_2 | P7_1 | P7_0 | E1VSS | E1VSS | E1VSS | E1VSS | E1VSS | E1VSS | P1_7 | VSS | | P1_3 | P1_2 | D |
| E | P4_4 | P4_5 | | P7_5 | VSS | VCC | E1VSS | E1VCC | E1VSS | E1VSS | E1VSS | E1VCC | E1VSS | E1VSS | VDD | VSS | E0VSS | | P1_1 | X1 | E |
| F | P4_6 | P4_7 | | E1VSS | VSS | | | | | | | | | | | SYSVCC | E0VSS | | OSCVSS | X2 | F |
| G | P4_8 | P4_9 | | E1VSS | VCC | | | VSS | VDD | VDD | VDD | VDD | VDD | | | E1VSS | E0VSS | | OSCVSS | RESETZ | G |
| H | P4_10 | P4_11 | | E1VSS | E1VSS | | VSS | | VSS | VSS | VSS | VSS | | VSS | | E1VCC | E0VSS | | JP0_4 | JP0_0 | H |
| J | P4_12 | P4_13 | | E1VSS | E1VCC | | VDD | VSS | | VSS | VSS | | VSS | VDD | | SYSVCC | E0VSS | | JP0_1 | JP0_2 | J |
| K | P4_14 | P6_0 | | A1VSS | VDD | | VDD | VSS | VSS | VSS | VSS | VSS | VSS | VDD | | E0VSS | P9_8 | | JP0_3 | JP0_5 | K |
| L | P6_1 | ADC1I0 | | A1VSS | VCC | | VDD | VSS | VSS | VSS | VSS | VSS | VSS | VDD | | P9_7 | E0VSS | | P0_14 | P0_13 | L |
| M | ADC1I1 | ADC1I2 | | A1VSS | VCC | | VDD | VSS | | VSS | VSS | | VSS | VDD | | E0VCC | E0VSS | | P0_10 | CVMOUTZ | M |
| N | ADC1I3 | ADC1I4 | | A1VSS | VSS | | VSS | | VSS | VSS | VSS | VSS | | VSS | | E0VSS | E0VSS | | FLMD0 | P0_9 | N |
| P | ADC1I5 | ADC1I6 | | A1VSS | A1VSS | | | VSS | VDD | VDD | VDD | VDD | VSS | | | E0VSS | E0VSS | | P0_8 | P0_7 | P |
| R | ADC1I7 | ADC1I8 | | A1VSS | A0VSS | | | | | | | | | | | VSS | E0VSS | | P0_6 | P0_5 | R |
| T | ADC1I9 | ADC1I10 | | A1VSS | A0VSS | A0VSS | A0VSS | A0VSS | A0VSS | E0VCC | E0VSS | E0VSS | E0VSS | P6_11 | VSS | VSS | E0VSS | | P0_4 | P0_3 | T |
| U | A1VCC | ADC1I11 | | A0VSS | A0VSS | A0VSS | A0VSS | A0VSS | A0VSS | P6_4 | P6_5 | P6_6 | E0VSS | P6_10 | P6_12 | P6_13 | VSS | | E0VSS | P0_2 | U |
| V | A1VREFH | A1VSS | | | | | | | | | | | | | | | | | E0VSS | P3_11 | V |
| W | A1VSS | A1VSS | A0VSS | ADC0I11 | ADC0I9 | ADC0I7 | ADC0I5 | ADC0I3 | ADC0I1 | P6_2 | P5_0 | P5_4 | P5_6 | P5_8 | P5_10 | P5_12 | P5_14 | P0_0 | P0_1 | VSS | W |
| Y | A1VSS | A0VSS | A0VREFH | A0VCC | ADC0I10 | ADC0I8 | ADC0I6 | ADC0I4 | ADC0I2 | ADC0I0 | P6_3 | P5_1 | P5_5 | P5_7 | P5_9 | P5_11 | P5_13 | P5_15 | VSS | VSS | Y |

**Figure 2.2 Pin Connection Diagram of P1M-C (BGA-292)**

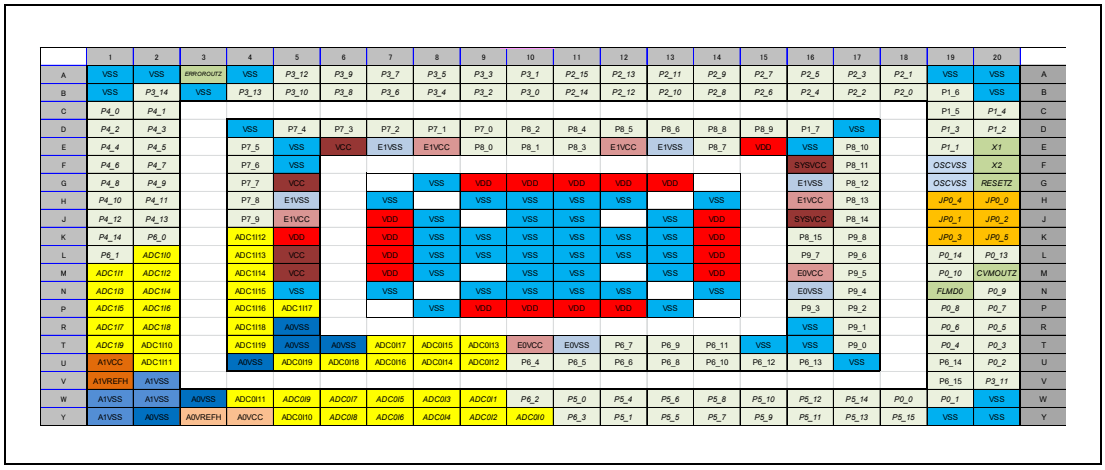
### 2.1.3 P1H-C (4MB) (BGA-292) (TOP View)



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | VSS | VSS | ERROROUTZ | VSS | P3_12 | P3_9 | P3_7 | P3_5 | P3_3 | P3_1 | P2_15 | P2_13 | P2_11 | P2_9 | P2_7 | P2_5 | P2_3 | P2_1 | VSS | VSS | A |
| B | VSS | P3_14 | VSS | P3_13 | P3_10 | P3_8 | P3_6 | P3_4 | P3_2 | P3_0 | P2_14 | P2_12 | P2_10 | P2_8 | P2_6 | P2_4 | P2_2 | P2_0 | P1_6 | VSS | B |
| C | P4_0 | P4_1 | | | | | | | | | | | | | | | | | P1_5 | P1_4 | C |
| D | P4_2 | P4_3 | | VSS | P7_4 | P7_3 | P7_2 | P7_1 | P7_0 | P8_2 | P8_4 | P8_5 | P8_6 | P8_8 | P8_9 | P1_7 | VSS | | P1_3 | P1_2 | D |
| E | P4_4 | P4_5 | | P7_5 | VSS | VCC | E1VSS | E1VCC | P8_0 | P8_1 | P8_3 | E1VCC | E1VSS | P8_7 | VDD | VSS | P8_10 | | P1_1 | X1 | E |
| F | P4_6 | P4_7 | | P7_6 | VSS | | | | | | | | | | | SYSVCC | P8_11 | | OSCVSS | X2 | F |
| G | P4_8 | P4_9 | | P7_7 | VCC | | | VSS | VDD | VDD | VDD | VDD | VDD | | | E1VSS | P8_12 | | OSCVSS | RESETZ | G |
| H | P4_10 | P4_11 | | P7_8 | E1VSS | | VSS | | VSS | VSS | VSS | VSS | | VSS | | E1VCC | P8_13 | | JP0_4 | JP0_0 | H |
| J | P4_12 | P4_13 | | P7_9 | E1VCC | | VDD | VSS | | VSS | VSS | | VSS | VDD | | SYSVCC | P8_14 | | JP0_1 | JP0_2 | J |
| K | P4_14 | P6_0 | | ADC1I12 | VDD | | VDD | VSS | VSS | VSS | VSS | VSS | VSS | VDD | | P8_15 | P9_8 | | JP0_3 | JP0_5 | K |
| L | P6_1 | ADC1I0 | | ADC1I13 | VCC | | VDD | VSS | VSS | VSS | VSS | VSS | VSS | VDD | | P9_7 | P9_6 | | P0_14 | P0_13 | L |
| M | ADC1I1 | ADC1I2 | | ADC1I14 | VCC | | VDD | VSS | | VSS | VSS | | VSS | VDD | | E0VCC | P9_5 | | P0_10 | CVMOUTZ | M |
| N | ADC1I3 | ADC1I4 | | ADC1I15 | VSS | | VSS | | VSS | VSS | VSS | VSS | | VSS | | E0VSS | P9_4 | | FLMD0 | P0_9 | N |
| P | ADC1I5 | ADC1I6 | | A1VSS | A1VSS | | | VSS | VDD | VDD | VDD | VDD | VSS | | | P9_3 | P9_2 | | P0_8 | P0_7 | P |
| R | ADC1I7 | ADC1I8 | | A1VSS | A0VSS | | | | | | | | | | | VSS | P9_1 | | P0_6 | P0_5 | R |
| T | ADC1I9 | ADC1I10 | | A1VSS | A0VSS | A0VSS | A0VSS | ADC0I15 | ADC0I13 | E0VCC | E0VSS | P6_7 | P6_9 | P6_11 | VSS | VSS | P9_0 | | P0_4 | P0_3 | T |
| U | A1VCC | ADC1I11 | | A0VSS | A0VSS | A0VSS | A0VSS | ADC0I14 | ADC0I12 | P6_4 | P6_5 | P6_6 | P6_8 | P6_10 | P6_12 | P6_13 | VSS | | P6_14 | P0_2 | U |
| V | A1VREFH | A1VSS | | | | | | | | | | | | | | | | | P6_15 | P3_11 | V |
| W | A1VSS | A1VSS | A0VSS | ADC0I11 | ADC0I9 | ADC0I7 | ADC0I5 | ADC0I3 | ADC0I1 | P6_2 | P5_0 | P5_4 | P5_6 | P5_8 | P5_10 | P5_12 | P5_14 | P0_0 | P0_1 | VSS | W |
| Y | A1VSS | A0VSS | A0VREFH | A0VCC | ADC0I10 | ADC0I8 | ADC0I6 | ADC0I4 | ADC0I2 | ADC0I0 | P6_3 | P5_1 | P5_5 | P5_7 | P5_9 | P5_11 | P5_13 | P5_15 | VSS | VSS | Y |

**Figure 2.3 Pin Connection Diagram of P1H-C (4MB) (BGA-292)**

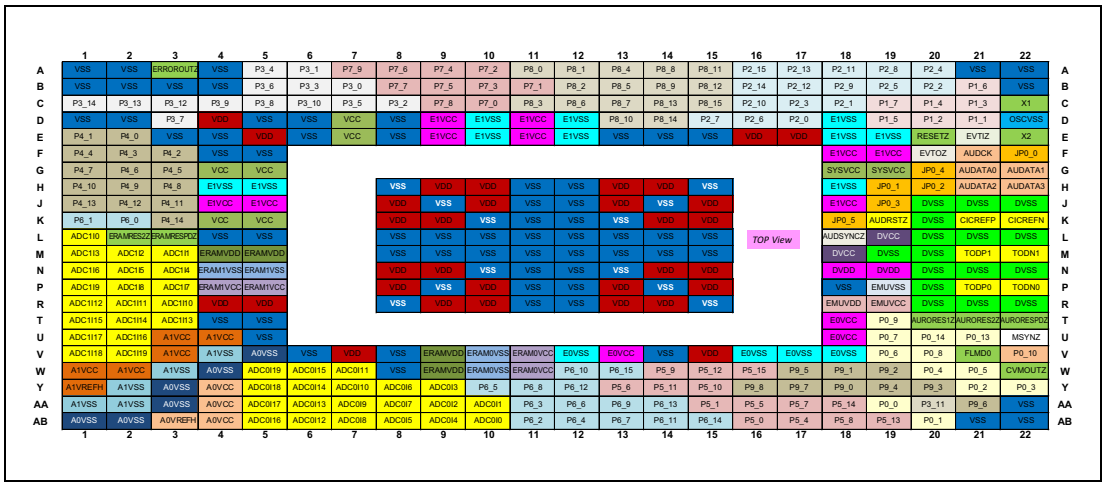
## 2.1.4 P1H-C (8MB) (BGA-292) (TOP View)



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | VSS | VSS | ERROROUT2 | VSS | P3_12 | P3_9 | P3_7 | P3_5 | P3_3 | P3_1 | P2_15 | P2_13 | P2_11 | P2_9 | P2_7 | P2_5 | P2_3 | P2_1 | VSS | VSS | A |
| B | VSS | P3_14 | VSS | P3_13 | P3_10 | P3_8 | P3_6 | P3_4 | P3_2 | P3_0 | P2_14 | P2_12 | P2_10 | P2_8 | P2_6 | P2_4 | P2_2 | P2_0 | P1_6 | VSS | B |
| C | P4_0 | P4_1 | | | | | | | | | | | | | | | | | P1_5 | P1_4 | C |
| D | P4_2 | P4_3 | | VSS | P7_4 | P7_3 | P7_2 | P7_1 | P7_0 | P8_2 | P8_4 | P8_5 | P8_6 | P8_8 | P8_9 | P1_7 | VSS | | P1_3 | P1_2 | D |
| E | P4_4 | P4_5 | | P7_5 | VSS | VCC | E1VSS | E1VCC | P8_0 | P8_1 | P8_3 | E1VCC | E1VSS | P8_7 | VDD | VSS | P8_10 | | P1_1 | X1 | E |
| F | P4_6 | P4_7 | | P7_6 | VSS | | | | | | | | | | | SYSVCC | P8_11 | | OSCVSS | X2 | F |
| G | P4_8 | P4_9 | | P7_7 | VCC | | | VSS | VDD | VDD | VDD | VDD | VDD | | | E1VSS | P8_12 | | OSCVSS | RESETZ | G |
| H | P4_10 | P4_11 | | P7_8 | E1VSS | | VSS | | VSS | VSS | VSS | VSS | | VSS | | E1VCC | P8_13 | | JP0_4 | JP0_0 | H |
| J | P4_12 | P4_13 | | P7_9 | E1VCC | | VDD | VSS | | VSS | VSS | | VSS | VDD | | SYSVCC | P8_14 | | JP0_1 | JP0_2 | J |
| K | P4_14 | P6_0 | | ADC1I12 | VDD | | VDD | VSS | VSS | VSS | VSS | VSS | VSS | VDD | | P8_15 | P9_8 | | JP0_3 | JP0_5 | K |
| L | P6_1 | ADC1I10 | | ADC1I13 | VCC | | VDD | VSS | VSS | VSS | VSS | VSS | VSS | VDD | | P9_7 | P9_6 | | P0_14 | P0_13 | L |
| M | ADC1I1 | ADC1I2 | | ADC1I14 | VCC | | VDD | VSS | | VSS | VSS | | VSS | VDD | | E0VCC | P9_5 | | P0_10 | CVMOUTZ | M |
| N | ADC1I3 | ADC1I4 | | ADC1I15 | VSS | | VSS | | VSS | VSS | VSS | VSS | | VSS | | E0VSS | P9_4 | | FLMD0 | P0_9 | N |
| P | ADC1I5 | ADC1I6 | | ADC1I16 | ADC1I17 | | | VSS | VDD | VDD | VDD | VDD | VSS | | | P9_3 | P9_2 | | P0_8 | P0_7 | P |
| R | ADC1I7 | ADC1I8 | | ADC1I18 | A0VSS | | | | | | | | | | | VSS | P9_1 | | P0_6 | P0_5 | R |
| T | ADC1I9 | ADC1I10 | | ADC1I19 | A0VSS | A0VSS | ADC0I17 | ADC0I15 | ADC0I13 | E0VCC | E0VSS | P6_7 | P6_9 | P6_11 | VSS | VSS | P9_0 | | P0_4 | P0_3 | T |
| U | A1VCC | ADC1I11 | | A0VSS | ADC0I19 | ADC0I18 | ADC0I16 | ADC0I14 | ADC0I12 | P6_4 | P6_5 | P6_6 | P6_8 | P6_10 | P6_12 | P6_13 | VSS | | P6_14 | P0_2 | U |
| V | A1VREFH | A1VSS | | | | | | | | | | | | | | | | | P6_15 | P3_11 | V |
| W | A1VSS | A1VSS | A0VSS | ADC0I11 | ADC0I9 | ADC0I7 | ADC0I5 | ADC0I3 | ADC0I1 | P6_2 | P5_0 | P5_4 | P5_6 | P5_8 | P5_10 | P5_12 | P5_14 | P0_0 | P0_1 | VSS | W |
| Y | A1VSS | A0VSS | A0VREFH | A0VCC | ADC0I10 | ADC0I8 | ADC0I6 | ADC0I4 | ADC0I2 | ADC0I0 | P6_3 | P5_1 | P5_5 | P5_7 | P5_9 | P5_11 | P5_13 | P5_15 | VSS | VSS | Y |

Figure 2.4 Pin Connection Diagram of P1H-C (8MB) (BGA-292)

## 2.1.5 P1H-CE (BGA-404) (TOP View)



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | VSS | VSS | ERROROUT2 | VSS | P3_4 | P3_1 | P7_9 | P7_6 | P7_4 | P7_2 | P8_0 | P8_1 | P8_4 | P8_8 | P8_11 | P2_15 | P2_13 | P2_11 | P2_8 | P2_4 | VSS | VSS | A |
| B | VSS | VSS | VSS | VSS | P3_6 | P3_3 | P3_0 | P7_7 | P7_5 | P7_3 | P7_1 | P8_2 | P8_5 | P8_9 | P8_12 | P2_14 | P2_12 | P2_9 | P2_5 | P2_2 | P1_6 | VSS | B |
| C | P3_14 | P3_13 | P3_12 | P3_9 | P3_8 | P3_10 | P3_5 | P3_2 | P7_8 | P7_0 | P8_3 | P8_6 | P8_7 | P8_13 | P8_15 | P2_10 | P2_3 | P2_1 | P1_7 | P1_4 | P1_3 | X1 | C |
| D | VSS | VSS | P3_7 | VDD | VSS | VSS | VCC | VSS | E1VCC | E1VSS | E1VCC | E1VSS | P8_10 | P8_14 | P2_7 | P2_6 | P2_0 | E1VSS | P1_5 | P1_2 | P1_1 | OSCVSS | D |
| E | P4_1 | P4_0 | VSS | VSS | VDD | VSS | VCC | VSS | E1VCC | E1VSS | E1VCC | E1VSS | VSS | VSS | VSS | VDD | VDD | E1VSS | E1VSS | RESETZ | EVTIZ | X2 | E |
| F | P4_4 | P4_3 | P4_2 | VSS | VSS | | | | | | | | | | | | | E1VCC | E1VCC | EVTOZ | AUDCK | JP0_0 | F |
| G | P4_7 | P4_6 | P4_5 | VCC | VCC | | | | | | | | | | | | | SYSVCC | SYSVCC | JP0_4 | AUDATA0 | AUDATA1 | G |
| H | P4_10 | P4_9 | P4_8 | E1VSS | E1VSS | | | VSS | VDD | VDD | VSS | VSS | VDD | VDD | VSS | | | E1VSS | JP0_1 | JP0_2 | AUDATA2 | AUDATA3 | H |
| J | P4_13 | P4_12 | P4_11 | E1VCC | E1VCC | | | VDD | VSS | VDD | VSS | VSS | VDD | VSS | VDD | | | E1VCC | JP0_3 | DVSS | DVSS | DVSS | J |
| K | P6_1 | P6_0 | P4_14 | VCC | VCC | | | VDD | VDD | VSS | VSS | VSS | VSS | VDD | VDD | | | JP0_5 | AUDRSTZ | DVSS | CICREFP | CICREFN | K |
| L | ADC1I0 | ERAMRESZ2 | ERAMRESPD2 | VSS | VSS | | | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | TOP View | | AUDSYNC2 | DVCC | DVSS | DVSS | DVSS | L |
| M | ADC1I3 | ADC1I2 | ADC1I1 | ERAMVDD | ERAMVDD | | | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | | | DVCC | DVSS | DVSS | TODP1 | TODN1 | M |
| N | ADC1I6 | ADC1I5 | ADC1I4 | ERAM1VSS | ERAM1VSS | | | VDD | VDD | VSS | VSS | VSS | VSS | VDD | VDD | | | DVDD | DVDD | DVSS | DVSS | DVSS | N |
| P | ADC1I9 | ADC1I8 | ADC1I7 | ERAM1VCC | ERAM1VCC | | | VDD | VSS | VDD | VSS | VSS | VDD | VSS | VDD | | | VSS | EMUVSS | DVSS | TODP0 | TODN0 | P |
| R | ADC1I12 | ADC1I11 | ADC1I10 | VDD | VDD | | | VSS | VDD | VDD | VSS | VSS | VDD | VDD | VSS | | | EMUVDD | EMUVCC | DVSS | DVSS | DVSS | R |
| T | ADC1I15 | ADC1I14 | ADC1I13 | VSS | VSS | | | | | | | | | | | | | E0VCC | P0_9 | AURORES1Z | AURORESZ2 | AURORESPD2 | T |
| U | ADC1I17 | ADC1I16 | A1VCC | A1VCC | VSS | | | | | | | | | | | | | E0VCC | P0_7 | P0_14 | P0_13 | MSYNZ | U |
| V | ADC1I18 | ADC1I19 | A1VCC | A1VSS | A0VSS | VSS | VDD | VSS | ERAMVDD | ERAM0VSS | ERAM0VCC | E0VSS | E0VCC | VSS | VDD | E0VSS | E0VSS | E0VSS | P0_6 | P0_8 | FLMD0 | P0_10 | V |
| W | A1VCC | A1VCC | A1VSS | A0VSS | ADC0I19 | ADC0I15 | ADC0I11 | VSS | ERAMVDD | ERAM0VSS | ERAM0VCC | P6_10 | P6_15 | P5_9 | P5_12 | P5_15 | P9_5 | P9_1 | P9_2 | P0_4 | P0_5 | CVMOUTZ | W |
| Y | A1VREFH | A1VSS | A0VSS | A0VCC | ADC0I18 | ADC0I14 | ADC0I10 | ADC0I6 | ADC0I3 | P6_5 | P6_8 | P6_12 | P5_6 | P5_11 | P5_10 | P9_8 | P9_7 | P9_0 | P9_4 | P9_3 | P0_2 | P0_3 | Y |
| AA | A1VSS | A1VSS | A0VSS | A0VCC | ADC0I17 | ADC0I13 | ADC0I9 | ADC0I7 | ADC0I2 | ADC0I1 | P6_3 | P6_6 | P6_9 | P6_13 | P5_1 | P5_5 | P5_7 | P5_14 | P0_0 | P3_11 | P9_6 | VSS | AA |
| AB | A0VSS | A0VSS | A0VREFH | A0VCC | ADC0I16 | ADC0I12 | ADC0I8 | ADC0I5 | ADC0I4 | ADC0I0 | P6_2 | P6_4 | P6_7 | P6_11 | P6_14 | P5_0 | P5_4 | P5_8 | P5_13 | P0_1 | VSS | VSS | AB |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | |

Figure 2.5 Pin Connection Diagram of P1H-CE (BGA-404)

### 2.1.6 P1M-C/P1H-C (BGA-156) (TOP View)

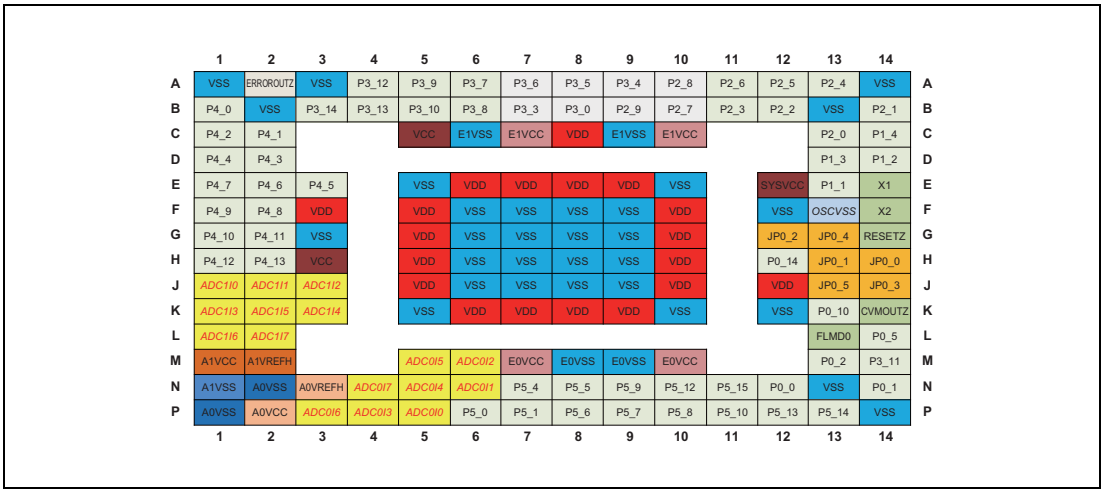



**Figure 2.6 Pin Connection Diagram of P1M-C/P1H-C (BGA-156)**

### 2.1.7 Term Definition

The following terms are used in this section:

Pin

Denotes the physical pin. Every pin is denoted by a unique pin number.
A pin can be used in several modes. Each pin is assigned a name that reflects its function, which is determined by the selected mode.

Port group

Denotes a group of pins. All the pins of a specific port group are controlled by the same port control register.
The RH850/P1x-C provides the following port groups, indicated by the numbers in the table below

| | P1M-C/P1H-C (156) | P1M-C (144) | P1M-C (292) | P1H-C (4MB) (292) | P1H-C (8MB) (292) | P1H-C (8MB) ED (404) |
|---|---|---|---|---|---|---|
| Number of Group | 7 | 8 | 10 | 11 | 11 | 11 |
| Name of Group | P0 to P5, JP0 | P0 to P6, JP0 | P0 to P7, P9, JP0 | P0 to P9, JP0 | P0 to P9, JP0 | P0 to P9, JP0 |

Port group index n

Each port group is identified by its own index “n” throughout this section; e.g.
PMCn for the port mode control register of the Pn port.

Port mode and ports

A pin in port mode works as a general purpose input/output pin. It is then called “port”.
The corresponding name is Pn_m. For example, P0_7 denotes port 7 of port group 0. It is referenced as “port P0_7”.

Alternative mode

In alternative mode, a pin can be used for various non-general-purpose input/output functions. It is such as CSI and INTP.

SHMT1/SHMT2/SHMT4

Denotes input buffer types. Each types have a different DC characteristics.
For detail, refer to the **Section 35, Electrical Specifications**.

GPIO/HSIO.

GPIO indicates General purpose I/O and HSIO indicates High speed I/O.
For detai, refer to the **Section 35, Electrical Specifications**.

# 2.2 Pin List

## 2.2.1 Pin List and Function assignment

**CAUTION**

**To prevent malfunction, secure reset value to registers that are not available for the individual product. For port availability of each product see Table 2.1.**

Table 2.1 Pin List (1/9)

| Ref. Nr. | P1M-C Pkg 144 0.4mm pin pitch | | P1M-C/P1H-C Pkg 156 0.8mm ball pitch | | P1M-C Pkg 292 0.8mm ball pitch | | P1H-C (4MB) Pkg 292 | | P1H-C (8MB) Pkg 292 | | P1H-CE Pkg 404 | | Superset Ref.Nr. | Port name | Port buffer settings Port Input Type | Port Input Type | Port Input Type | Port Input Type | Port Output Type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | CMOS | SHMT1 | SHMT4 | SHMT2 | |
| 1 | | | | | | | | | | | | | 1 | | | | | | |
| 2 | 1 | P3_14 | B3 | P3_14 | B2 | P3_14 | B2 | P3_14 | B2 | P3_14 | C1 | P3_14 | 2 | P3_14 | | SHMT1 | SHMT4 | | HSIO |
| 3 | 2 | P4_0 | B1 | P4_0 | C1 | P4_0 | C1 | P4_0 | C1 | P4_0 | E2 | P4_0 | 3 | P4_0 | | SHMT1 | SHMT4 | | GPIO |
| 4 | 3 | P4_1 | C2 | P4_1 | C2 | P4_1 | C2 | P4_1 | C2 | P4_1 | E1 | P4_1 | 4 | P4_1 | | SHMT1 | SHMT4 | | GPIO |
| 5 | 4 | P4_2 | C1 | P4_2 | D1 | P4_2 | D1 | P4_2 | D1 | P4_2 | F3 | P4_2 | 5 | P4_2 | | SHMT1 | SHMT4 | | GPIO |
| 6 | | | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | 6 | VDD | | | | | |
| 7 | 5 | P4_3 | D2 | P4_3 | D2 | P4_3 | D2 | P4_3 | D2 | P4_3 | F2 | P4_3 | 7 | P4_3 | | SHMT1 | SHMT4 | | GPIO |
| 8 | 6 | P4_4 | D1 | P4_4 | E1 | P4_4 | E1 | P4_4 | E1 | P4_4 | F1 | P4_4 | 8 | P4_4 | | SHMT1 | SHMT4 | | GPIO |
| 9 | 7 | P4_5 | E3 | P4_5 | E2 | P4_5 | E2 | P4_5 | E2 | P4_5 | G3 | P4_5 | 9 | P4_5 | | SHMT1 | SHMT4 | | GPIO |
| 10 | 8 | P4_6 | E2 | P4_6 | F1 | P4_6 | F1 | P4_6 | F1 | P4_6 | G2 | P4_6 | 10 | P4_6 | | SHMT1 | SHMT4 | | GPIO |
| 11 | 9 | P4_7 | E1 | P4_7 | F2 | P4_7 | F2 | P4_7 | F2 | P4_7 | G1 | P4_7 | 11 | P4_7 | | SHMT1 | SHMT4 | | GPIO |
| 12 | 10 | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | 12 | E1VSS | | | | | |
| 13 | 11 | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | 13 | E1VCC | | | | | |
| 14 | 12 | P4_8 | F2 | P4_8 | G1 | P4_8 | G1 | P4_8 | G1 | P4_8 | H3 | P4_8 | 14 | P4_8 | | SHMT1 | SHMT4 | | GPIO |
| 15 | 13 | P4_9 | F1 | P4_9 | G2 | P4_9 | G2 | P4_9 | G2 | P4_9 | H2 | P4_9 | 15 | P4_9 | | SHMT1 | SHMT4 | | GPIO |
| 16 | 14 | P4_10 | G1 | P4_10 | H1 | P4_10 | H1 | P4_10 | H1 | P4_10 | H1 | P4_10 | 16 | P4_10 | | SHMT1 | SHMT4 | | GPIO |
| 17 | 15 | P4_11 | G2 | P4_11 | H2 | P4_11 | H2 | P4_11 | H2 | P4_11 | J3 | P4_11 | 17 | P4_11 | | SHMT1 | SHMT4 | | GPIO |
| 18 | 16 | P4_12 | H1 | P4_12 | J1 | P4_12 | J1 | P4_12 | J1 | P4_12 | J2 | P4_12 | 18 | P4_12 | | SHMT1 | SHMT4 | | GPIO |
| 19 | 17 | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | 19 | VDD | | | | | |
| 20 | 18 | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | 20 | VSS | | | | | |
| 21 | 19 | VCC | VCC | VCC | VCC | VCC | VCC | VCC | VCC | VCC | VCC | VCC | 21 | VCC | | | | | |
| 22 | 20 | P4_13 | H2 | P4_13 | J2 | P4_13 | J2 | P4_13 | J2 | P4_13 | J1 | P4_13 | 22 | P4_13 | | SHMT1 | SHMT4 | | GPIO |
| 23 | 21 | P4_14 | | | K1 | P4_14 | K1 | P4_14 | K1 | P4_14 | K3 | P4_14 | 23 | P4_14 | | SHMT1 | SHMT4 | | GPIO |
| 24 | 22 | P6_0 | | | K2 | P6_0 | K2 | P6_0 | K2 | P6_0 | K2 | P6_0 | 24 | P6_0 | | SHMT1 | SHMT4 | | GPIO |
| 25 | 23 | P6_1 | | | L1 | P6_1 | L1 | P6_1 | L1 | P6_1 | K1 | P6_1 | 25 | P6_1 | | SHMT1 | SHMT4 | | GPIO |
| 26 | | | | | | | | | | | L3 | ERAM RESPDZ | 26 | ERAM RESPDZ | | | | SHMT2 | |
| 27 | | | | | | | | | | | L2 | ERAM RES2Z | 27 | ERAM RES2Z | | | | SHMT2 | |
| 28 | 24 | ADC1I0 | J1 | ADC1I0 | L2 | ADC1I0 | L2 | ADC1I0 | L2 | ADC1I0 | L1 | ADC1I0 | 28 | ADC1I0 | | | | | |
| 29 | 25 | ADC1I1 | J2 | ADC1I1 | M1 | ADC1I1 | M1 | ADC1I1 | M1 | ADC1I1 | M3 | ADC1I1 | 29 | ADC1I1 | | | | | |
| 30 | 26 | ADC1I2 | J3 | ADC1I2 | M2 | ADC1I2 | M2 | ADC1I2 | M2 | ADC1I2 | M2 | ADC1I2 | 30 | ADC1I2 | | | | | |
| 31 | 27 | ADC1I3 | K1 | ADC1I3 | N1 | ADC1I3 | N1 | ADC1I3 | N1 | ADC1I3 | M1 | ADC1I3 | 31 | ADC1I3 | | | | | |
| 32 | 28 | ADC1I4 | K3 | ADC1I4 | N2 | ADC1I4 | N2 | ADC1I4 | N2 | ADC1I4 | N3 | ADC1I4 | 32 | ADC1I4 | | | | | |
| 33 | 29 | ADC1I5 | K2 | ADC1I5 | P1 | ADC1I5 | P1 | ADC1I5 | P1 | ADC1I5 | N2 | ADC1I5 | 33 | ADC1I5 | | | | | |
| 34 | 30 | ADC1I6 | L1 | ADC1I6 | P2 | ADC1I6 | P2 | ADC1I6 | P2 | ADC1I6 | N1 | ADC1I6 | 34 | ADC1I6 | | | | | |
| 35 | 31 | ADC1I7 | L2 | ADC1I7 | R1 | ADC1I7 | R1 | ADC1I7 | R1 | ADC1I7 | P3 | ADC1I7 | 35 | ADC1I7 | | | | | |
| 36 | 32 | ADC1I8 | | | R2 | ADC1I8 | R2 | ADC1I8 | R2 | ADC1I8 | P2 | ADC1I8 | 36 | ADC1I8 | | | | | |
| 37 | 33 | ADC1I9 | | | T1 | ADC1I9 | T1 | ADC1I9 | T1 | ADC1I9 | P1 | ADC1I9 | 37 | ADC1I9 | | | | | |
| 38 | | | | | T2 | ADC1I10 | T2 | ADC1I10 | T2 | ADC1I10 | R3 | ADC1I10 | 38 | ADC1I10 | | | | | |
| 39 | | | | | U2 | ADC1I11 | U2 | ADC1I11 | U2 | ADC1I11 | R2 | ADC1I11 | 39 | ADC1I11 | | | | | |
| 40 | | | | | K4 | A1VSS | K4 | ADC1I12 | K4 | ADC1I12 | R1 | ADC1I12 | 40 | ADC1I12 | | | | | |
| 41 | | | | | L4 | A1VSS | L4 | ADC1I13 | L4 | ADC1I13 | T3 | ADC1I13 | 41 | ADC1I13 | | | | | |
| 42 | | | | | M4 | A1VSS | M4 | ADC1I14 | M4 | ADC1I14 | T2 | ADC1I14 | 42 | ADC1I14 | | | | | |
| 43 | | | | | N4 | A1VSS | N4 | ADC1I15 | N4 | ADC1I15 | T1 | ADC1I15 | 43 | ADC1I15 | | | | | |
| 44 | | | | | P4 | A1VSS | P4 | A1VSS | P4 | ADC1I16 | U2 | ADC1I16 | 44 | ADC1I16 | | | | | |
| 45 | | | | | P5 | A1VSS | P5 | A1VSS | P5 | ADC1I17 | U1 | ADC1I17 | 45 | ADC1I17 | | | | | |

## Table 2.1 Pin List (2/9)

| Ref. Nr. | P1M-C Pkg 144 0.4mm pin pitch | | P1M-C/P1H-C Pkg 156 0.8mm ball pitch | | P1M-C Pkg 292 0.8mm ball pitch | | P1H-C (4MB) Pkg 292 | | P1H-C (8MB) Pkg 292 | | P1H-CE Pkg 404 | | Superset Ref.Nr. | Port name | Port buffer settings Port Input Type | Port Input Type | Port Input Type | Port Input Type | Port Output Type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 46 | | | | | R4 | A1VSS | R4 | A1VSS | R4 | ADC1I18 | V1 | ADC1I18 | 46 | ADC1I18 | | | | | |
| 47 | | | | | T4 | A1VSS | T4 | A1VSS | T4 | ADC1I19 | V2 | ADC1I19 | 47 | ADC1I19 | | | | | |
| 48 | 34 | A1VCC | M1 | A1VCC | U1 | A1VCC | U1 | A1VCC | U1 | A1VCC | A1 VCC | A1VCC | 48 | A1VCC | | | | | |
| 49 | 35 | A1VREFH | M2 | A1VREFH | V1 | A1VREFH | V1 | A1VREFH | V1 | A1VREFH | Y1 | A1VREFH | 49 | A1VREFH | | | | | |
| 50 | 36 | A1VSS | N1 | A1VSS | V2 | A1VSS | V2 | A1VSS | V2 | A1VSS | Y2 | A1VSS | 50 | A1VSS | | | | | |
| 51 | 36 | A1VSS | A1 VSS | A1VSS | A1 VSS | A1VSS | A1 VSS | A1VSS | A1 VSS | A1VSS | A1 VSS | A1VSS | 51 | A1VSS | | | | | |
| 52 | 48 | | | | | | | | | | | | 52 | | | | | | |
| 53 | 37 | A0VREFH | N3 | A0VREFH | Y3 | A0VREFH | Y3 | A0VREFH | Y3 | A0VREFH | AB3 | A0VREFH | 53 | A0VREFH | | | | | |
| 54 | 38 | A0VSS | N2 | A0VSS | W3 | A0VSS | W3 | A0VSS | W3 | A0VSS | AA3 | A0VSS | 54 | A0VSS | | | | | |
| 55 | 38 | A0VSS | A0 VSS | A0VSS | A0 VSS | A0VSS | A0 VSS | A0VSS | A0 VSS | A0VSS | A0 VSS | A0VSS | 55 | A0VSS | | | | | |
| 56 | 39 | A0VCC | P2 | A0VCC | Y4 | A0VCC | Y4 | A0VCC | Y4 | A0VCC | A0 VCC | A0VCC | 56 | A0VCC | | | | | |
| 57 | | | | | U5 | A0VSS | U5 | A0VSS | U5 | ADC0I19 | W5 | ADC0I19 | 57 | ADC0I19 | | | | | |
| 58 | | | | | U6 | A0VSS | U6 | A0VSS | U6 | ADC0I18 | Y5 | ADC0I18 | 58 | ADC0I18 | | | | | |
| 59 | | | | | T7 | A0VSS | T7 | A0VSS | T7 | ADC0I17 | AA5 | ADC0I17 | 59 | ADC0I17 | | | | | |
| 60 | | | | | U7 | A0VSS | U7 | A0VSS | U7 | ADC0I16 | AB5 | ADC0I16 | 60 | ADC0I16 | | | | | |
| 61 | | | | | T8 | A0VSS | T8 | ADC0I15 | T8 | ADC0I15 | W6 | ADC0I15 | 61 | ADC0I15 | | | | | |
| 62 | | | | | U8 | A0VSS | U8 | ADC0I14 | U8 | ADC0I14 | Y6 | ADC0I14 | 62 | ADC0I14 | | | | | |
| 63 | | | | | T9 | A0VSS | T9 | ADC0I13 | T9 | ADC0I13 | AA6 | ADC0I13 | 63 | ADC0I13 | | | | | |
| 64 | | | | | U9 | A0VSS | U9 | ADC0I12 | U9 | ADC0I12 | AB6 | ADC0I12 | 64 | ADC0I12 | | | | | |
| 65 | | | | | W4 | ADC0I11 | W4 | ADC0I11 | W4 | ADC0I11 | W7 | ADC0I11 | 65 | ADC0I11 | | | | | |
| 66 | | | | | Y5 | ADC0I10 | Y5 | ADC0I10 | Y5 | ADC0I10 | Y7 | ADC0I10 | 66 | ADC0I10 | | | | | |
| 67 | 40 | ADC0I9 | | | W5 | ADC0I9 | W5 | ADC0I9 | W5 | ADC0I9 | AA7 | ADC0I9 | 67 | ADC0I9 | | | | | |
| 68 | 41 | ADC0I8 | | | Y6 | ADC0I8 | Y6 | ADC0I8 | Y6 | ADC0I8 | AB7 | ADC0I8 | 68 | ADC0I8 | | | | | |
| 69 | 42 | ADC0I7 | N4 | ADC0I7 | W6 | ADC0I7 | W6 | ADC0I7 | W6 | ADC0I7 | AA8 | ADC0I7 | 69 | ADC0I7 | | | | | |
| 70 | 43 | ADC0I6 | P3 | ADC0I6 | Y7 | ADC0I6 | Y7 | ADC0I6 | Y7 | ADC0I6 | Y8 | ADC0I6 | 70 | ADC0I6 | | | | | |
| 71 | 44 | ADC0I5 | M5 | ADC0I5 | W7 | ADC0I5 | W7 | ADC0I5 | W7 | ADC0I5 | AB8 | ADC0I5 | 71 | ADC0I5 | | | | | |
| 72 | 45 | ADC0I4 | N5 | ADC0I4 | Y8 | ADC0I4 | Y8 | ADC0I4 | Y8 | ADC0I4 | AB9 | ADC0I4 | 72 | ADC0I4 | | | | | |
| 73 | 46 | ADC0I3 | P4 | ADC0I3 | W8 | ADC0I3 | W8 | ADC0I3 | W8 | ADC0I3 | Y9 | ADC0I3 | 73 | ADC0I3 | | | | | |
| 74 | 47 | ADC0I2 | M6 | ADC0I2 | Y9 | ADC0I2 | Y9 | ADC0I2 | Y9 | ADC0I2 | AA9 | ADC0I2 | 74 | ADC0I2 | | | | | |
| 75 | 48 | ADC0I1 | N6 | ADC0I1 | W9 | ADC0I1 | W9 | ADC0I1 | W9 | ADC0I1 | AA10 | ADC0I1 | 75 | ADC0I1 | | | | | |
| 76 | 49 | ADC0I0 | P5 | ADC0I0 | Y10 | ADC0I0 | Y10 | ADC0I0 | Y10 | ADC0I0 | AB10 | ADC0I0 | 76 | ADC0I0 | | | | | |
| 77 | | | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | 77 | VDD | | | | | |
| 78 | 50 | P6_2 | | | W10 | P6_2 | W10 | P6_2 | W10 | P6_2 | AB11 | P6_2 | 78 | P6_2 | | SHMT1 | SHMT4 | | GPIO |
| 79 | 51 | P6_3 | | | Y11 | P6_3 | Y11 | P6_3 | Y11 | P6_3 | AA11 | P6_3 | 79 | P6_3 | | SHMT1 | SHMT4 | | GPIO |
| 80 | | | | | U10 | P6_4 | U10 | P6_4 | U10 | P6_4 | AB12 | P6_4 | 80 | P6_4 | | SHMT1 | SHMT4 | | GPIO |
| 81 | | | | | U11 | P6_5 | U11 | P6_5 | U11 | P6_5 | Y10 | P6_5 | 81 | P6_5 | | SHMT1 | SHMT4 | | GPIO |
| 82 | | | | | U12 | P6_6 | U12 | P6_6 | U12 | P6_6 | AA12 | P6_6 | 82 | P6_6 | | SHMT1 | SHMT4 | | GPIO |
| 83 | | | | | E0 VCC | E0VCC | E0 VCC | E0VCC | E0 VCC | E0VCC | E0 VCC | E0VCC | 83 | E0VCC | | | | | |
| 84 | | | | | E0 VSS | E0VSS | E0 VSS | E0VSS | E0 VSS | E0VSS | E0 VSS | E0VSS | 84 | E0VSS | | | | | |
| 85 | | | | | T12 | E0VSS | T12 | P6_7 | T12 | P6_7 | AB13 | P6_7 | 85 | P6_7 | | SHMT1 | SHMT4 | | GPIO |
| 86 | | | | | U13 | E0VSS | U13 | P6_8 | U13 | P6_8 | Y11 | P6_8 | 86 | P6_8 | | SHMT1 | SHMT4 | | GPIO |
| 87 | | | | | T13 | E0VSS | T13 | P6_9 | T13 | P6_9 | AA13 | P6_9 | 87 | P6_9 | | SHMT1 | SHMT4 | | GPIO |
| 88 | | | | | U14 | P6_10 | U14 | P6_10 | U14 | P6_10 | W12 | P6_10 | 88 | P6_10 | | SHMT1 | SHMT4 | | GPIO |
| 89 | | | | | T14 | P6_11 | T14 | P6_11 | T14 | P6_11 | AB14 | P6_11 | 89 | P6_11 | | SHMT1 | SHMT4 | | GPIO |
| 90 | | | | | U15 | P6_12 | U15 | P6_12 | U15 | P6_12 | Y12 | P6_12 | 90 | P6_12 | | SHMT1 | SHMT4 | | GPIO |
| 91 | | | | | U16 | P6_13 | U16 | P6_13 | U16 | P6_13 | AA14 | P6_13 | 91 | P6_13 | | SHMT1 | SHMT4 | | GPIO |
| 92 | | | | | U19 | E0VSS | U19 | P6_14 | U19 | P6_14 | AB15 | P6_14 | 92 | P6_14 | | SHMT1 | SHMT4 | | GPIO |
| 93 | | | | | V19 | E0VSS | V19 | P6_15 | V19 | P6_15 | W13 | P6_15 | 93 | P6_15 | | SHMT1 | SHMT4 | | GPIO |
| 94 | 52 | P5_0 | P6 | P5_0 | W11 | P5_0 | W11 | P5_0 | W11 | P5_0 | AB16 | P5_0 | 94 | P5_0 | | SHMT1 | SHMT4 | | GPIO |
| 95 | 53 | P5_1 | P7 | P5_1 | Y12 | P5_1 | Y12 | P5_1 | Y12 | P5_1 | AA15 | P5_1 | 95 | P5_1 | | SHMT1 | SHMT4 | | GPIO |
| 96 | 54 | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | 96 | VSS | | | | | |
| 97 | 55 | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | 97 | VDD | | | | | |
| 98 | 56 | P5_4 | N7 | P5_4 | W12 | P5_4 | W12 | P5_4 | W12 | P5_4 | AB17 | P5_4 | 98 | P5_4 | | SHMT1 | SHMT4 | | GPIO |
| 99 | 57 | P5_5 | N8 | P5_5 | Y13 | P5_5 | Y13 | P5_5 | Y13 | P5_5 | AA16 | P5_5 | 99 | P5_5 | | SHMT1 | SHMT4 | | GPIO |
| 100 | 58 | P5_6 | P8 | P5_6 | W13 | P5_6 | W13 | P5_6 | W13 | P5_6 | Y13 | P5_6 | 100 | P5_6 | | SHMT1 | SHMT4 | | GPIO |
| 101 | 59 | P5_7 | P9 | P5_7 | Y14 | P5_7 | Y14 | P5_7 | Y14 | P5_7 | AA17 | P5_7 | 101 | P5_7 | | SHMT1 | SHMT4 | | GPIO |

## Table 2.1 Pin List (3/9)

| Ref. Nr. | P1M-C Pkg 144 0.4mm pin pitch | | P1M-C/P1H-C Pkg 156 0.8mm ball pitch | | P1M-C Pkg 292 0.8mm ball pitch | | P1H-C (4MB) Pkg 292 | | P1H-C (8MB) Pkg 292 | | P1H-CE Pkg 404 | | Superset Ref.Nr. | Port name | Port buffer settings: Port Input Type | Port Input Type | Port Input Type | Port Input Type | Port Output Type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 102 | 60 | E0VCC | E0 VCC | E0VCC | E0 VCC | E0VCC | E0 VCC | E0VCC | E0 VCC | E0VCC | E0 VCC | E0VCC | 102 | E0VCC | | | | | |
| 103 | 61 | E0VSS | E0 VSS | E0VSS | E0 VSS | E0VSS | E0 VSS | E0VSS | E0 VSS | E0VSS | E0 VSS | E0VSS | 103 | E0VSS | | | | | |
| 104 | 62 | P5_8 | P10 | P5_8 | W14 | P5_8 | W14 | P5_8 | W14 | P5_8 | AB18 | P5_8 | 104 | P5_8 | | SHMT1 | SHMT4 | | GPIO |
| 105 | 63 | P5_9 | N9 | P5_9 | Y15 | P5_9 | Y15 | P5_9 | Y15 | P5_9 | W14 | P5_9 | 105 | P5_9 | | SHMT1 | SHMT4 | | GPIO |
| 106 | 64 | P5_10 | P11 | P5_10 | W15 | P5_10 | W15 | P5_10 | W15 | P5_10 | Y15 | P5_10 | 106 | P5_10 | | SHMT1 | SHMT4 | | GPIO |
| 107 | 65 | P5_11 | | | Y16 | P5_11 | Y16 | P5_11 | Y16 | P5_11 | Y14 | P5_11 | 107 | P5_11 | | SHMT1 | SHMT4 | | GPIO |
| 108 | 66 | P5_12 | N10 | P5_12 | W16 | P5_12 | W16 | P5_12 | W16 | P5_12 | W15 | P5_12 | 108 | P5_12 | | SHMT1 | SHMT4 | | GPIO |
| 109 | 67 | P5_13 | P12 | P5_13 | Y17 | P5_13 | Y17 | P5_13 | Y17 | P5_13 | AB19 | P5_13 | 109 | P5_13 | | SHMT1 | SHMT4 | | GPIO |
| 110 | 68 | P5_14 | P13 | P5_14 | W17 | P5_14 | W17 | P5_14 | W17 | P5_14 | AA18 | P5_14 | 110 | P5_14 | | SHMT1 | SHMT4 | | GPIO |
| 111 | 69 | P5_15 | N11 | P5_15 | Y18 | P5_15 | Y18 | P5_15 | Y18 | P5_15 | W16 | P5_15 | 111 | P5_15 | | SHMT1 | SHMT4 | | GPIO |
| 112 | 70 | P0_0 | N12 | P0_0 | W18 | P0_0 | W18 | P0_0 | W18 | P0_0 | AA19 | P0_0 | 112 | P0_0 | | SHMT1 | SHMT4 | | GPIO |
| 113 | 71 | P0_1 | N14 | P0_1 | W19 | P0_1 | W19 | P0_1 | W19 | P0_1 | AB20 | P0_1 | 113 | P0_1 | | SHMT1 | SHMT4 | | GPIO |
| 114 | 72 | P3_11 | M14 | P3_11 | V20 | P3_11 | V20 | P3_11 | V20 | P3_11 | AA20 | P3_11 | 114 | P3_11 | | SHMT1 | SHMT4 | | GPIO |
| 115 | | | | | T17 | E0VSS | T17 | P9_0 | T17 | P9_0 | Y18 | P9_0 | 115 | P9_0 | | SHMT1 | SHMT4 | | GPIO |
| 116 | | | | | R17 | E0VSS | R17 | P9_1 | R17 | P9_1 | W18 | P9_1 | 116 | P9_1 | | SHMT1 | SHMT4 | | GPIO |
| 117 | | | | | P17 | E0VSS | P17 | P9_2 | P17 | P9_2 | W19 | P9_2 | 117 | P9_2 | | SHMT1 | SHMT4 | | GPIO |
| 118 | | | | | P16 | E0VSS | P16 | P9_3 | P16 | P9_3 | Y20 | P9_3 | 118 | P9_3 | | SHMT1 | SHMT4 | | GPIO |
| 119 | | | E0 VCC | E0VCC | E0 VCC | E0VCC (see comment) | E0 VCC | E0VCC | E0 VCC | E0VCC | E0 VCC | E0VCC | 119 | E0VCC | | | | | |
| 120 | | | E0 VSS | E0VSS | E0 VSS | E0VSS (see comment) | E0 VSS | E0VSS | E0 VSS | E0VSS | E0 VSS | E0VSS | 120 | E0VSS | | | | | |
| 121 | | | | | N17 | E0VSS | N17 | P9_4 | N17 | P9_4 | Y19 | P9_4 | 121 | P9_4 | | SHMT1 | SHMT4 | | GPIO |
| 122 | | | | | M17 | E0VSS | M17 | P9_5 | M17 | P9_5 | W17 | P9_5 | 122 | P9_5 | | SHMT1 | SHMT4 | | GPIO |
| 123 | | | | | L17 | E0VSS | L17 | P9_6 | L17 | P9_6 | AA21 | P9_6 | 123 | P9_6 | | SHMT1 | SHMT4 | | GPIO |
| 124 | | | | | L16 | P9_7 (see comment) | L16 | P9_7 | L16 | P9_7 | Y17 | P9_7 | 124 | P9_7 | | SHMT1 | SHMT4 | | GPIO |
| 125 | | | | | K17 | P9_8 (see comment) | K17 | P9_8 | K17 | P9_8 | Y16 | P9_8 | 125 | P9_8 | | SHMT1 | SHMT4 | | GPIO |
| 126 | | | | | | 66 | | | | | | | 126 | | | | | | |
| 127 | 73 | P0_2 | M13 | P0_2 | U20 | P0_2 | U20 | P0_2 | U20 | P0_2 | Y21 | P0_2 | 127 | P0_2 | | SHMT1 | SHMT4 | | GPIO |
| 128 | 74 | P0_3 | | | T20 | P0_3 | T20 | P0_3 | T20 | P0_3 | Y22 | P0_3 | 128 | P0_3 | | SHMT1 | SHMT4 | | GPIO |
| 129 | 75 | P0_4 | | | T19 | P0_4 | T19 | P0_4 | T19 | P0_4 | W20 | P0_4 | 129 | P0_4 | | SHMT1 | SHMT4 | | GPIO |
| 130 | 76 | P0_5 | L14 | P0_5 | R20 | P0_5 | R20 | P0_5 | R20 | P0_5 | W21 | P0_5 | 130 | P0_5 | | SHMT1 | SHMT4 | | GPIO |
| 131 | 77 | P0_6 | | | R19 | P0_6 | R19 | P0_6 | R19 | P0_6 | V19 | P0_6 | 131 | P0_6 | | SHMT1 | SHMT4 | | GPIO |
| 132 | 78 | P0_7 | | | P20 | P0_7 | P20 | P0_7 | P20 | P0_7 | U19 | P0_7 | 132 | P0_7 | | SHMT1 | SHMT4 | | GPIO |
| 133 | 79 | P0_8 | | | P19 | P0_8 | P19 | P0_8 | P19 | P0_8 | V20 | P0_8 | 133 | P0_8 | | SHMT1 | SHMT4 | | GPIO |
| 134 | 80 | P0_9 | | | N20 | P0_9 | N20 | P0_9 | N20 | P0_9 | T19 | P0_9 | 134 | P0_9 | | SHMT1 | SHMT4 | | GPIO |
| 135 | 81 | FLMD0 | L13 | FLMD0 | N19 | FLMD0 | N19 | FLMD0 | N19 | FLMD0 | V21 | FLMD0 | 135 | FLMD0 | | | | SHMT2 | |
| 136 | 82 | CVM OUTZ | K14 | CVM OUTZ | M20 | CVM OUTZ | M20 | CVMOUTZ | M20 | CVMOUTZ | W22 | CVM OUTZ | 136 | CVM OUTZ | | | | SHMT2 | GPIO50O hm (CVM OUT) |
| 137 | 83 | E0VSS | E0 VSS | E0VSS | E0 VSS | E0VSS | E0 VSS | E0VSS | E0 VSS | E0VSS | E0 VSS | E0VSS | 137 | E0VSS | | | | | |
| 138 | 84 | E0VCC | E0 VCC | E0VCC | E0 VCC | E0VCC | E0 VCC | E0VCC | E0 VCC | E0VCC | E0 VCC | E0VCC | 138 | E0VCC | | | | | |
| 139 | 85 | P0_10 | K13 | P0_10 | M19 | P0_10 | M19 | P0_10 | M19 | P0_10 | V22 | P0_10 | 139 | P0_10 | | SHMT1 | SHMT4 | | GPIO (RESETOUT) |
| 140 | 86 | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | 140 | VDD | | | | | |
| 141 | 87 | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | 141 | VSS | | | | | |
| 142 | 88 | P0_13 | | | L20 | P0_13 | L20 | P0_13 | L20 | P0_13 | U21 | P0_13 | 142 | P0_13 | | SHMT1 | SHMT4 | | GPIO |
| 143 | 89 | P0_14 | H12 | P0_14 | L19 | P0_14 | L19 | P0_14 | L19 | P0_14 | U20 | P0_14 | 143 | P0_14 | | SHMT1 | SHMT4 | | GPIO |
| 144 | | | | | | | | | | | U22 | MSYNZ | 144 | MSYNZ | | SHMT1 | | | |
| 145 | | | | | | | | | | | R19 | EMUVCC | 145 | EMUVCC | | | | | |
| 146 | | | | | | | | | | | P19 | EMUVSS | 146 | EMUVSS | | | | | |
| 147 | | | | | | | | | | | T21 | AURO RES2Z | 147 | AURO RES2Z | | | | SHMT2 | |
| 148 | | | | | | | | | | | T20 | AURO RES1Z | 148 | AURO RES1Z | | | | SHMT2 | |
| 149 | | | | | | | | | | | T22 | AURO RESPDZ | 149 | AURO RESPDZ | | | | SHMT2 | |
| 150 | | | | | | | | | | | R18 | EMUVDD | 150 | EMUVDD | | | | | |
| 151 | | | | | | | | | | | | EMUVDD | 151 | EMUVDD | | | | | |
| 152 | | | | | | | | | | | P18 | VSS | 152 | VSS | | | | | |
| 153 | | | | | | | | | | | | VSS | 153 | VSS | | | | | |

## Table 2.1 Pin List (4/9)

| Ref. Nr. | P1M-C Pkg 144 0.4mm pin pitch | | P1M-C/P1H-C Pkg 156 0.8mm ball pitch | | P1M-C Pkg 292 0.8mm ball pitch | | P1H-C (4MB) Pkg 292 | | P1H-C (8MB) Pkg 292 | | P1H-CE Pkg 404 | | Superset Ref.Nr. | Port name | Port buffer settings Port Input Type | Port Input Type | Port Input Type | Port Input Type | Port Output Type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 154 | | | | | | | | | | | N18 | DVDD | 154 | DVDD | | | | | |
| 155 | | | | | | | | | | | P21 | TODP0 | 155 | TODP0 | | | | | |
| 156 | | | | | | | | | | | P22 | TODN0 | 156 | TODN0 | | | | | |
| 157 | | | | | | | | | | | R20 | DVSS | 157 | DVSS | | | | | |
| 158 | | | | | | | | | | | M21 | TODP1 | 158 | TODP1 | | | | | |
| 159 | | | | | | | | | | | M22 | TODN1 | 159 | TODN1 | | | | | |
| 160 | | | | | | | | | | | N19 | DVDD | 160 | DVDD | | | | | |
| 161 | | | | | | | | | | | N20 | DVSS | 161 | DVSS | | | | | |
| 162 | | | | | | | | | | | M19 | DVSS | 162 | DVSS | | | | | |
| 164 | | | | | | | | | | | K21 | CICREFP | 164 | CICREFP | | | | | |
| 165 | | | | | | | | | | | K22 | CICREFN | 165 | CICREFN | | | | | |
| 166 | | | | | | | | | | | L19 | DVCC | 166 | DVCC | | | | | |
| 167 | | | | | | | | | | | M18 | DVCC | 167 | DVCC | | | | | |
| 168 | | | | | | | | | | | M19 | DVSS | 168 | DVSS | | | | | |
| 169 | | | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | 169 | VSS | | | | | |
| 170 | 90 | JP0_5 | J13 | JP0_5 | K20 | JP0_5 | K20 | JP0_5 | K20 | JP0_5 | K18 | JP0_5 | 170 | JP0_5 | | SHMT1 | SHMT4 | | GPIO |
| 171 | 91 | JP0_3 | J14 | JP0_3 | K19 | JP0_3 | K19 | JP0_3 | K19 | JP0_3 | J19 | JP0_3 | 171 | JP0_3 | | SHMT1 | SHMT4 | | GPIO |
| 172 | 92 | JP0_2 | G12 | JP0_2 | J20 | JP0_2 | J20 | JP0_2 | J20 | JP0_2 | H20 | JP0_2 | 172 | JP0_2 | | SHMT1 | SHMT4 | | GPIO |
| 175 | 93 | JP0_1 | H13 | JP0_1 | J19 | JP0_1 | J19 | JP0_1 | J19 | JP0_1 | H19 | JP0_1 | 175 | JP0_1 | | SHMT1 | SHMT4 | | GPIO |
| 176 | 94 | JP0_0 | H14 | JP0_0 | H20 | JP0_0 | H20 | JP0_0 | H20 | JP0_0 | F22 | JP0_0 | 176 | JP0_0 | | SHMT1 | SHMT4 | | GPIO |
| 177 | 95 | JP0_4 | G13 | JP0_4 | H19 | JP0_4 | H19 | JP0_4 | H19 | JP0_4 | G20 | JP0_4 | 177 | JP0_4 | | | | SHMT2 | |
| 178 | | | | | | | | | | | F20 | EVTOZ | 178 | EVTOZ | | | | | GPIO50 Ohm |
| 179 | | | | | | | | | | | E21 | EVTIZ | 179 | EVTIZ | | SHMT1 | | | |
| 182 | | | | | | | | | | | H22 | AUDATA3 | 182 | AUDATA3 | CMOS | | | | |
| 183 | | | | | | | | | | | H21 | AUDATA2 | 183 | AUDATA2 | CMOS | | | | |
| 184 | | | | | | | | | | | G22 | AUDATA1 | 184 | AUDATA1 | CMOS | | | | |
| 185 | | | | | | | | | | | G21 | AUDATA0 | 185 | AUDATA0 | CMOS | | | | |
| 186 | | | | | | | | | | | L18 | AUD SYNCZ | 186 | AUD SYNCZ | CMOS | | | | |
| 187 | | | | | | | | | | | F21 | AUDCK | 187 | AUDCK | CMOS | | | | |
| 188 | | | | | | | | | | | K19 | AUDRSTZ | 188 | AUDRSTZ | CMOS | | | | |
| 189 | 96 | RESETZ | G14 | RESETZ | G20 | RESETZ | G20 | RESETZ | G20 | RESETZ | E20 | RESETZ | 189 | RESETZ | | | | SHMT2 | |
| 190 | 97 | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | 190 | VSS | | | | | |
| 191 | 98 | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | 191 | VDD | | | | | |
| 192 | 99 | X2 | F14 | X2 | F20 | X2 | F20 | X2 | F20 | X2 | E22 | X2 | 192 | X2 | | | | | |
| 193 | 100 | OSCVSS | F13 | OSCVSS | F19 | OSCVSS | F19 | OSCVSS | F19 | OSCVSS | D22 | OSCVSS | 193 | OSCVSS | | | | | |
| 194 | | | | | G19 | OSCVSS | G19 | OSCVSS | G19 | OSCVSS | | | 194 | OSCVSS | | | | | |
| 195 | 101 | X1 | E14 | X1 | E20 | X1 | E20 | X1 | E20 | X1 | C22 | X1 | 195 | X1 | Special | | | | |
| 196 | 102 | SYSVCC | SYS VCC | SYSVCC | SYS VCC | SYSVCC | SYS VCC | SYSVCC | SYS VCC | SYSVCC | SYS VCC | SYSVCC | 196 | SYSVCC | | | | | |
| 197 | 103 | P1_1 | E13 | P1_1 | E19 | P1_1 | E19 | P1_1 | E19 | P1_1 | D21 | P1_1 | 197 | P1_1 | | SHMT1 | SHMT4 | | GPIO |
| 198 | 104 | P1_2 | D14 | P1_2 | D20 | P1_2 | D20 | P1_2 | D20 | P1_2 | D20 | P1_2 | 198 | P1_2 | | SHMT1 | SHMT4 | | GPIO |
| 199 | 105 | P1_3 | D13 | P1_3 | D19 | P1_3 | D19 | P1_3 | D19 | P1_3 | C21 | P1_3 | 199 | P1_3 | | SHMT1 | SHMT4 | | GPIO |
| 200 | 106 | P1_4 | C14 | P1_4 | C20 | P1_4 | C20 | P1_4 | C20 | P1_4 | C20 | P1_4 | 200 | P1_4 | | SHMT1 | SHMT4 | | GPIO |
| 201 | 107 | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | 201 | E1VCC | | | | | |
| 202 | 108 | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | 202 | E1VSS | | | | | |
| 203 | 45 | | | | | | | | | | | | 203 | | | | | | |
| 204 | | | | | C19 | P1_5 | C19 | P1_5 | C19 | P1_5 | D19 | P1_5 | 204 | P1_5 | | SHMT1 | SHMT4 | | GPIO |
| 205 | | | | | B19 | P1_6 | B19 | P1_6 | B19 | P1_6 | B21 | P1_6 | 205 | P1_6 | | SHMT1 | SHMT4 | | GPIO |
| 206 | | | | | D16 | P1_7 | D16 | P1_7 | D16 | P1_7 | C19 | P1_7 | 206 | P1_7 | | SHMT1 | SHMT4 | | GPIO |
| 207 | 109 | P2_0 | C13 | P2_0 | B18 | P2_0 | B18 | P2_0 | B18 | P2_0 | D17 | P2_0 | 207 | P2_0 | | SHMT1 | SHMT4 | | GPIO |
| 208 | 110 | P2_1 | B14 | P2_1 | A18 | P2_1 | A18 | P2_1 | A18 | P2_1 | C18 | P2_1 | 208 | P2_1 | | SHMT1 | SHMT4 | | GPIO |
| 209 | 111 | P2_2 | B12 | P2_2 | B17 | P2_2 | B17 | P2_2 | B17 | P2_2 | B20 | P2_2 | 209 | P2_2 | | SHMT1 | SHMT4 | | GPIO |
| 210 | 112 | P2_3 | B11 | P2_3 | A17 | P2_3 | A17 | P2_3 | A17 | P2_3 | C17 | P2_3 | 210 | P2_3 | | SHMT1 | SHMT4 | | GPIO |
| 211 | 113 | P2_4 | A13 | P2_4 | B16 | P2_4 | B16 | P2_4 | B16 | P2_4 | A20 | P2_4 | 211 | P2_4 | | SHMT1 | SHMT4 | | GPIO |
| 212 | 114 | P2_5 | A12 | P2_5 | A16 | P2_5 | A16 | P2_5 | A16 | P2_5 | B19 | P2_5 | 212 | P2_5 | | SHMT1 | SHMT4 | | GPIO |
| 213 | 115 | P2_6 | A11 | P2_6 | B15 | P2_6 | B15 | P2_6 | B15 | P2_6 | D16 | P2_6 | 213 | P2_6 | | SHMT1 | SHMT4 | | GPIO |
| 214 | 116 | P2_7 | B10 | P2_7 | A15 | P2_7 | A15 | P2_7 | A15 | P2_7 | D15 | P2_7 | 214 | P2_7 | | SHMT1 | SHMT4 | | GPIO |

## Table 2.1 Pin List (5/9)

| Ref. Nr. | P1M-C Pkg 144 0.4mm pin pitch | | P1M-C/P1H-C Pkg 156 0.8mm ball pitch | | P1M-C Pkg 292 0.8mm ball pitch | | P1H-C (4MB) Pkg 292 | | P1H-C (8MB) Pkg 292 | | P1H-CE Pkg 404 | | Superset Ref.Nr. | Port name | Port buffer settings Port Input Type | Port Input Type | Port Input Type | Port Input Type | Port Output Type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 215 | 117 | P2_8 | A10 | P2_8 | B14 | P2_8 | B14 | P2_8 | B14 | P2_8 | A19 | P2_8 | 215 | P2_8 | | SHMT1 | SHMT4 | | GPIO |
| 216 | 118 | P2_9 | B9 | P2_9 | A14 | P2_9 | A14 | P2_9 | A14 | P2_9 | B18 | P2_9 | 216 | P2_9 | | SHMT1 | SHMT4 | | GPIO |
| 217 | 119 | P2_10 | | | B13 | P2_10 | B13 | P2_10 | B13 | P2_10 | C16 | P2_10 | 217 | P2_10 | | SHMT1 | SHMT4 | | GPIO |
| 218 | 120 | P2_11 | | | A13 | P2_11 | A13 | P2_11 | A13 | P2_11 | A18 | P2_11 | 218 | P2_11 | | SHMT1 | SHMT4 | | GPIO |
| 219 | 121 | P2_12 | | | B12 | P2_12 | B12 | P2_12 | B12 | P2_12 | B17 | P2_12 | 219 | P2_12 | | SHMT1 | SHMT4 | | GPIO |
| 220 | 122 | P2_13 | | | A12 | P2_13 | A12 | P2_13 | A12 | P2_13 | A17 | P2_13 | 220 | P2_13 | | SHMT1 | SHMT4 | | GPIO |
| 221 | 123 | P2_14 | | | B11 | P2_14 | B11 | P2_14 | B11 | P2_14 | B16 | P2_14 | 221 | P2_14 | | SHMT1 | SHMT4 | | GPIO |
| 222 | 124 | VSS | | | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | 222 | VSS | | | | | |
| 223 | 125 | VDD | | | VDD | VDD | VDD | VDD | VDD | VDD | VDD | VDD | 223 | VDD | | | | | |
| 224 | 126 | P2_15 | | | A11 | P2_15 | A11 | P2_15 | A11 | P2_15 | A16 | P2_15 | 224 | P2_15 | | SHMT1 | SHMT4 | | GPIO |
| 225 | | | | | K16 | E1VSS | K16 | P8_15 | K16 | P8_15 | C15 | P8_15 | 225 | P8_15 | | SHMT1 | SHMT4 | | GPIO |
| 226 | | | | | J17 | E1VSS | J17 | P8_14 | J17 | P8_14 | D14 | P8_14 | 226 | P8_14 | | SHMT1 | SHMT4 | | GPIO |
| 227 | | | | | H17 | E1VSS | H17 | P8_13 | H17 | P8_13 | C14 | P8_13 | 227 | P8_13 | | SHMT1 | SHMT4 | | GPIO |
| 228 | | | | | G17 | E1VSS | G17 | P8_12 | G17 | P8_12 | B15 | P8_12 | 228 | P8_12 | | SHMT1 | SHMT4 | | GPIO |
| 229 | | | | | F17 | E1VSS | F17 | P8_11 | F17 | P8_11 | A15 | P8_11 | 229 | P8_11 | | SHMT1 | SHMT4 | | HSIO |
| 230 | | | | | E17 | E1VSS | E17 | P8_10 | E17 | P8_10 | D13 | P8_10 | 230 | P8_10 | | SHMT1 | SHMT4 | | HSIO |
| 231 | | | | | D15 | E1VSS | D15 | P8_9 | D15 | P8_9 | B14 | P8_9 | 231 | P8_9 | | SHMT1 | SHMT4 | | HSIO |
| 232 | | | | | D14 | E1VSS | D14 | P8_8 | D14 | P8_8 | A14 | P8_8 | 232 | P8_8 | | SHMT1 | SHMT4 | | GPIO |
| 233 | | | | | E14 | E1VSS | E14 | P8_7 | E14 | P8_7 | C13 | P8_7 | 233 | P8_7 | | SHMT1 | SHMT4 | | GPIO |
| 234 | | | | | D13 | E1VSS | D13 | P8_6 | D13 | P8_6 | C12 | P8_6 | 234 | P8_6 | | SHMT1 | SHMT4 | | GPIO |
| 235 | | | | | E1 VCC | E1VSS | E1 VCC | E1VSS | E1 VCC | E1VSS | E1 VCC | E1VSS | 235 | E1VSS | | | | | |
| 236 | | | | | E1 VSS | E1VSS | E1 VSS | E1VCC | E1 VSS | E1VCC | E1 VSS | E1VCC | 236 | E1VCC | | | | | |
| 237 | | | | | D12 | E1VSS | D12 | P8_5 | D12 | P8_5 | B13 | P8_5 | 237 | P8_5 | | SHMT1 | SHMT4 | | GPIO |
| 238 | | | | | D11 | E1VSS | D11 | P8_4 | D11 | P8_4 | A13 | P8_4 | 238 | P8_4 | | SHMT1 | SHMT4 | | GPIO |
| 239 | | | | | E11 | E1VSS | E11 | P8_3 | E11 | P8_3 | C11 | P8_3 | 239 | P8_3 | | SHMT1 | SHMT4 | | GPIO |
| 240 | | | | | D10 | E1VSS | D10 | P8_2 | D10 | P8_2 | B12 | P8_2 | 240 | P8_2 | | SHMT1 | SHMT4 | | GPIO |
| 241 | | | | | E10 | E1VSS | E10 | P8_1 | E10 | P8_1 | A12 | P8_1 | 241 | P8_1 | | SHMT1 | SHMT4 | | GPIO |
| 242 | | | | | E9 | E1VSS | E9 | P8_0 | E9 | P8_0 | A11 | P8_0 | 242 | P8_0 | | SHMT1 | SHMT4 | | GPIO |
| 243 | | | | | D9 | P7_0 | D9 | P7_0 | D9 | P7_0 | C10 | P7_0 | 243 | P7_0 | | SHMT1 | SHMT4 | | GPIO |
| 244 | | | | | D8 | P7_1 | D8 | P7_1 | D8 | P7_1 | B11 | P7_1 | 244 | P7_1 | | SHMT1 | SHMT4 | | GPIO |
| 245 | | | | | D7 | P7_2 | D7 | P7_2 | D7 | P7_2 | A10 | P7_2 | 245 | P7_2 | | SHMT1 | SHMT4 | | GPIO |
| 246 | | | | | D6 | P7_3 | D6 | P7_3 | D6 | P7_3 | B10 | P7_3 | 246 | P7_3 | | SHMT1 | SHMT4 | | GPIO |
| 247 | | | | | D5 | P7_4 | D5 | P7_4 | D5 | P7_4 | A9 | P7_4 | 247 | P7_4 | | SHMT1 | SHMT4 | | GPIO |
| 248 | | | | | E4 | P7_5 | E4 | P7_5 | E4 | P7_5 | B9 | P7_5 | 248 | P7_5 | | SHMT1 | SHMT4 | | GPIO |
| 249 | | | | | F4 | E1VSS | F4 | P7_6 | F4 | P7_6 | A8 | P7_6 | 249 | P7_6 | | SHMT1 | SHMT4 | | GPIO |
| 250 | | | | | G4 | E1VSS | G4 | P7_7 | G4 | P7_7 | B8 | P7_7 | 250 | P7_7 | | SHMT1 | SHMT4 | | GPIO |
| 251 | | | | | H4 | E1VSS | H4 | P7_8 | H4 | P7_8 | C9 | P7_8 | 251 | P7_8 | | SHMT1 | SHMT4 | | GPIO |
| 252 | | | | | J4 | E1VSS | J4 | P7_9 | J4 | P7_9 | A7 | P7_9 | 252 | P7_9 | | SHMT1 | SHMT4 | | GPIO |
| 253 | 127 | P3_0 | B8 | P3_0 | B10 | P3_0 | B10 | P3_0 | B10 | P3_0 | B7 | P3_0 | 253 | P3_0 | | SHMT1 | SHMT4 | | GPIO |
| 254 | 128 | P3_1 | | | A10 | P3_1 | A10 | P3_1 | A10 | P3_1 | A6 | P3_1 | 254 | P3_1 | | SHMT1 | SHMT4 | | GPIO |
| 255 | 129 | P3_2 | | | B9 | P3_2 | B9 | P3_2 | B9 | P3_2 | C8 | P3_2 | 255 | P3_2 | | SHMT1 | SHMT4 | | GPIO |
| 256 | 130 | P3_3 | B7 | P3_3 | A9 | P3_3 | A9 | P3_3 | A9 | P3_3 | B6 | P3_3 | 256 | P3_3 | | SHMT1 | SHMT4 | | GPIO |
| 257 | 131 | P3_4 | A9 | P3_4 | B8 | P3_4 | B8 | P3_4 | B8 | P3_4 | A5 | P3_4 | 257 | P3_4 | | SHMT1 | SHMT4 | | GPIO |
| 258 | 132 | P3_5 | A8 | P3_5 | A8 | P3_5 | A8 | P3_5 | A8 | P3_5 | C7 | P3_5 | 258 | P3_5 | | SHMT1 | SHMT4 | | GPIO |
| 259 | 133 | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | E1 VCC | E1VCC | 259 | E1VCC | | | | | |
| 260 | 134 | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | E1 VSS | E1VSS | 260 | E1VSS | | | | | |
| 261 | 135 | P3_6 | A7 | P3_6 | B7 | P3_6 | B7 | P3_6 | B7 | P3_6 | B5 | P3_6 | 261 | P3_6 | | SHMT1 | SHMT4 | | GPIO |
| 262 | 136 | P3_7 | A6 | P3_7 | A7 | P3_7 | A7 | P3_7 | A7 | P3_7 | D3 | P3_7 | 262 | P3_7 | | SHMT1 | SHMT4 | | GPIO |
| 263 | 137 | P3_8 | B6 | P3_8 | B6 | P3_8 | B6 | P3_8 | B6 | P3_8 | C5 | P3_8 | 263 | P3_8 | | SHMT1 | SHMT4 | | GPIO |
| 264 | 138 | P3_9 | A5 | P3_9 | A6 | P3_9 | A6 | P3_9 | A6 | P3_9 | C4 | P3_9 | 264 | P3_9 | | SHMT1 | SHMT4 | | HSIO |
| 265 | 139 | P3_10 | B5 | P3_10 | B5 | P3_10 | B5 | P3_10 | B5 | P3_10 | C6 | P3_10 | 265 | P3_10 | | SHMT1 | SHMT4 | | HSIO |
| 266 | 140 | P3_12 | A4 | P3_12 | A5 | P3_12 | A5 | P3_12 | A5 | P3_12 | C3 | P3_12 | 266 | P3_12 | | SHMT1 | SHMT4 | | GPIO |
| 267 | 141 | P3_13 | B4 | P3_13 | B4 | P3_13 | B4 | P3_13 | B4 | P3_13 | C2 | P3_13 | 267 | P3_13 | | SHMT1 | SHMT4 | | GPIO |
| 268 | 142 | VDD | VDD | VDD | VDD | VDD | | | | | | | 268 | VDD | | | | | |
| 269 | 143 | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | VSS | 269 | VSS | | | | | |
| 270 | 144 | ERROR OUTZ | A2 | ERROR OUTZ | A3 | ERROR OUTZ | A3 | ERROR OUTZ | A3 | ERROR OUTZ | A3 | ERROR OUTZ | 270 | ERROR OUTZ | | | | SHMT2 | GPIO50 Ohm (ERROR OUT) |

## Table 2.1 Pin List (6/9)

| Ref. Nr. | P1M-C Pkg 144 0.4mm pin pitch | | P1M-C/P1H-C Pkg 156 0.8mm ball pitch | | P1M-C Pkg 292 0.8mm ball pitch | | P1H-C (4MB) Pkg 292 | | P1H-C (8MB) Pkg 292 | | P1H-CE Pkg 404 | | Superset Ref.Nr. | Port name | Port buffer settings: Port Input Type | Port Input Type | Port Input Type | Port Input Type | Port Output Type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 271 | | | | | | | | | | | | | 271 | | | | | | |
| 272 | | | E12 | SYSVCC | F16 | SYSVCC | F16 | SYSVCC | F16 | SYSVCC | G18 | SYSVCC | 272 | SYSVCC | | | | | |
| 273 | | | | | J16 | SYSVCC | J16 | SYSVCC | J16 | SYSVCC | G19 | SYSVCC | 273 | SYSVCC | | | | | |
| 274 | | | C5 | VCC | G5 | VCC | G5 | VCC | G5 | VCC | D7 | VCC | 274 | VCC | | | | | |
| 275 | | | H3 | VCC | L5 | VCC | L5 | VCC | L5 | VCC | E7 | VCC | 275 | VCC | | | | | |
| 276 | | | | | M5 | VCC | M5 | VCC | M5 | VCC | G4 | VCC | 276 | VCC | | | | | |
| 277 | | | | | E6 | VCC | E6 | VCC | E6 | VCC | G5 | VCC | 277 | VCC | | | | | |
| 278 | | | | | | | | | | | K4 | VCC | 278 | VCC | | | | | |
| 279 | | | | | | | | | | | K5 | VCC | 279 | VCC | | | | | |
| 280 | | | C8 | VDD | K5 | VDD | K5 | VDD | K5 | VDD | D4 | VDD | 280 | VDD | | | | | |
| 281 | | | E6 | VDD | E15 | VDD | E15 | VDD | E15 | VDD | E5 | VDD | 281 | VDD | | | | | |
| 282 | | | E7 | VDD | J7 | VDD | J7 | VDD | J7 | VDD | E16 | VDD | 282 | VDD | | | | | |
| 283 | | | E8 | VDD | K7 | VDD | K7 | VDD | K7 | VDD | E17 | VDD | 283 | VDD | | | | | |
| 284 | | | E9 | VDD | L7 | VDD | L7 | VDD | L7 | VDD | H9 | VDD | 284 | VDD | | | | | |
| 285 | | | F3 | VDD | M7 | VDD | M7 | VDD | M7 | VDD | H10 | VDD | 285 | VDD | | | | | |
| 286 | | | F5 | VDD | P9 | VDD | P9 | VDD | P9 | VDD | H13 | VDD | 286 | VDD | | | | | |
| 287 | | | F10 | VDD | P10 | VDD | P10 | VDD | P10 | VDD | H14 | VDD | 287 | VDD | | | | | |
| 288 | | | G5 | VDD | P11 | VDD | P11 | VDD | P11 | VDD | J8 | VDD | 288 | VDD | | | | | |
| 289 | | | G10 | VDD | P12 | VDD | P12 | VDD | P12 | VDD | J10 | VDD | 289 | VDD | | | | | |
| 290 | | | H5 | VDD | M14 | VDD | M14 | VDD | M14 | VDD | J13 | VDD | 290 | VDD | | | | | |
| 291 | | | H10 | VDD | L14 | VDD | L14 | VDD | L14 | VDD | J15 | VDD | 291 | VDD | | | | | |
| 292 | | | J5 | VDD | K14 | VDD | K14 | VDD | K14 | VDD | K8 | VDD | 292 | VDD | | | | | |
| 293 | | | J10 | VDD | J14 | VDD | J14 | VDD | J14 | VDD | K9 | VDD | 293 | VDD | | | | | |
| 294 | | | J12 | VDD | G13 | VDD | G13 | VDD | G13 | VDD | K14 | VDD | 294 | VDD | | | | | |
| 295 | | | K6 | VDD | G12 | VDD | G12 | VDD | G12 | VDD | K15 | VDD | 295 | VDD | | | | | |
| 296 | | | K7 | VDD | G11 | VDD | G11 | VDD | G11 | VDD | N8 | VDD | 296 | VDD | | | | | |
| 297 | | | K8 | VDD | G10 | VDD | G10 | VDD | G10 | VDD | N9 | VDD | 297 | VDD | | | | | |
| 298 | | | K9 | VDD | G9 | VDD | G9 | VDD | G9 | VDD | N14 | VDD | 298 | VDD | | | | | |
| 299 | | | | | | | | | | | N15 | VDD | 299 | VDD | | | | | |
| 300 | | | | | | | | | | | P8 | VDD | 300 | VDD | | | | | |
| 301 | | | | | | | | | | | P10 | VDD | 301 | VDD | | | | | |
| 302 | | | | | | | | | | | P13 | VDD | 302 | VDD | | | | | |
| 303 | | | | | | | | | | | P15 | VDD | 303 | VDD | | | | | |
| 304 | | | | | | | | | | | R4 | VDD | 304 | VDD | | | | | |
| 305 | | | | | | | | | | | R5 | VDD | 305 | VDD | | | | | |
| 306 | | | | | | | | | | | R9 | VDD | 306 | VDD | | | | | |
| 307 | | | | | | | | | | | R10 | VDD | 307 | VDD | | | | | |
| 308 | | | | | | | | | | | R13 | VDD | 308 | VDD | | | | | |
| 309 | | | | | | | | | | | R14 | VDD | 309 | VDD | | | | | |
| 310 | | | | | | | | | | | V7 | VDD | 310 | VDD | | | | | |
| 311 | | | | | | | | | | | V15 | VDD | 311 | VDD | | | | | |
| 312 | | | A1 | VSS | A1 | VSS | A1 | VSS | A1 | VSS | A1 | VSS | 312 | VSS | | | | | |
| 313 | | | A3 | VSS | A2 | VSS | A2 | VSS | A2 | VSS | A2 | VSS | 313 | VSS | | | | | |
| 314 | | | B2 | VSS | A4 | VSS | A4 | VSS | A4 | VSS | A4 | VSS | 314 | VSS | | | | | |
| 315 | | | P14 | VSS | B1 | VSS | B1 | VSS | B1 | VSS | A21 | VSS | 315 | VSS | | | | | |
| 316 | | | N13 | VSS | B3 | VSS | B3 | VSS | B3 | VSS | A22 | VSS | 316 | VSS | | | | | |
| 317 | | | A14 | VSS | Y19 | VSS | Y19 | VSS | Y19 | VSS | B1 | VSS | 317 | VSS | | | | | |
| 318 | | | B13 | VSS | Y20 | VSS | Y20 | VSS | Y20 | VSS | B2 | VSS | 318 | VSS | | | | | |
| 319 | | | E5 | VSS | W20 | VSS | W20 | VSS | W20 | VSS | B3 | VSS | 319 | VSS | | | | | |
| 320 | | | E10 | VSS | B20 | VSS | B20 | VSS | B20 | VSS | B4 | VSS | 320 | VSS | | | | | |
| 321 | | | F6 | VSS | A20 | VSS | A20 | VSS | A20 | VSS | B22 | VSS | 321 | VSS | | | | | |
| 322 | | | F7 | VSS | A19 | VSS | A19 | VSS | A19 | VSS | D1 | VSS | 322 | VSS | | | | | |
| 323 | | | F8 | VSS | D4 | VSS | D4 | VSS | D4 | VSS | D2 | VSS | 323 | VSS | | | | | |
| 324 | | | F9 | VSS | E5 | VSS | E5 | VSS | E5 | VSS | D5 | VSS | 324 | VSS | | | | | |
| 325 | | | F12 | VSS | F5 | VSS | F5 | VSS | F5 | VSS | D6 | VSS | 325 | VSS | | | | | |
| 326 | | | G3 | VSS | N5 | VSS | N5 | VSS | N5 | VSS | D8 | VSS | 326 | VSS | | | | | |
| 327 | | | G6 | VSS | U17 | VSS | U17 | VSS | U17 | VSS | E3 | VSS | 327 | VSS | | | | | |
| 328 | | | G7 | VSS | T16 | VSS | T16 | VSS | T16 | VSS | E4 | VSS | 328 | VSS | | | | | |
| 329 | | | G8 | VSS | T15 | VSS | T15 | VSS | T15 | VSS | E6 | VSS | 329 | VSS | | | | | |

## Table 2.1 Pin List (7/9)

| Ref. Nr. | P1M-C Pkg 144 0.4mm pin pitch | | P1M-C/P1H-C Pkg 156 0.8mm ball pitch | | P1M-C Pkg 292 0.8mm ball pitch | | P1H-C (4MB) Pkg 292 | | P1H-C (8MB) Pkg 292 | | P1H-CE Pkg 404 | | Superset Ref.Nr. | Port name | Port buffer settings Port Input Type | Port Input Type | Port Input Type | Port Input Type | Port Output Type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 330 | | | G9 | VSS | R16 | VSS | R16 | VSS | R16 | VSS | E8 | VSS | 330 | VSS | | | | | |
| 331 | | | H6 | VSS | E16 | VSS | E16 | VSS | E16 | VSS | E13 | VSS | 331 | VSS | | | | | |
| 332 | | | H7 | VSS | D17 | VSS | D17 | VSS | D17 | VSS | E14 | VSS | 332 | VSS | | | | | |
| 333 | | | H8 | VSS | G8 | VSS | G8 | VSS | G8 | VSS | E15 | VSS | 333 | VSS | | | | | |
| 334 | | | H9 | VSS | H7 | VSS | H7 | VSS | H7 | VSS | F4 | VSS | 334 | VSS | | | | | |
| 335 | | | J6 | VSS | H9 | VSS | H9 | VSS | H9 | VSS | F5 | VSS | 335 | VSS | | | | | |
| 336 | | | J7 | VSS | H10 | VSS | H10 | VSS | H10 | VSS | H8 | VSS | 336 | VSS | | | | | |
| 337 | | | J8 | VSS | H11 | VSS | H11 | VSS | H11 | VSS | H11 | VSS | 337 | VSS | | | | | |
| 338 | | | J9 | VSS | H12 | VSS | H12 | VSS | H12 | VSS | H12 | VSS | 338 | VSS | | | | | |
| 339 | | | K5 | VSS | H14 | VSS | H14 | VSS | H14 | VSS | H15 | VSS | 339 | VSS | | | | | |
| 340 | | | K10 | VSS | J8 | VSS | J8 | VSS | J8 | VSS | J9 | VSS | 340 | VSS | | | | | |
| 341 | | | K12 | VSS | J10 | VSS | J10 | VSS | J10 | VSS | J11 | VSS | 341 | VSS | | | | | |
| 342 | | | | | J11 | VSS | J11 | VSS | J11 | VSS | J12 | VSS | 342 | VSS | | | | | |
| 343 | | | | | J13 | VSS | J13 | VSS | J13 | VSS | J14 | VSS | 343 | VSS | | | | | |
| 344 | | | | | K8 | VSS | K8 | VSS | K8 | VSS | K10 | VSS | 344 | VSS | | | | | |
| 345 | | | | | K9 | VSS | K9 | VSS | K9 | VSS | K11 | VSS | 345 | VSS | | | | | |
| 346 | | | | | K10 | VSS | K10 | VSS | K10 | VSS | K12 | VSS | 346 | VSS | | | | | |
| 347 | | | | | K11 | VSS | K11 | VSS | K11 | VSS | K13 | VSS | 347 | VSS | | | | | |
| 348 | | | | | K12 | VSS | K12 | VSS | K12 | VSS | L8 | VSS | 348 | VSS | | | | | |
| 349 | | | | | K13 | VSS | K13 | VSS | K13 | VSS | L9 | VSS | 349 | VSS | | | | | |
| 350 | | | | | L8 | VSS | L8 | VSS | L8 | VSS | L10 | VSS | 350 | VSS | | | | | |
| 351 | | | | | L9 | VSS | L9 | VSS | L9 | VSS | L11 | VSS | 351 | VSS | | | | | |
| 352 | | | | | L10 | VSS | L10 | VSS | L10 | VSS | L12 | VSS | 352 | VSS | | | | | |
| 353 | | | | | L11 | VSS | L11 | VSS | L11 | VSS | L13 | VSS | 353 | VSS | | | | | |
| 354 | | | | | L12 | VSS | L12 | VSS | L12 | VSS | L14 | VSS | 354 | VSS | | | | | |
| 355 | | | | | L13 | VSS | L13 | VSS | L13 | VSS | L15 | VSS | 355 | VSS | | | | | |
| 356 | | | | | M8 | VSS | M8 | VSS | M8 | VSS | M8 | VSS | 356 | VSS | | | | | |
| 357 | | | | | M10 | VSS | M10 | VSS | M10 | VSS | M9 | VSS | 357 | VSS | | | | | |
| 358 | | | | | M11 | VSS | M11 | VSS | M11 | VSS | M10 | VSS | 358 | VSS | | | | | |
| 359 | | | | | M13 | VSS | M13 | VSS | M13 | VSS | M11 | VSS | 359 | VSS | | | | | |
| 360 | | | | | N7 | VSS | N7 | VSS | N7 | VSS | M12 | VSS | 360 | VSS | | | | | |
| 361 | | | | | N9 | VSS | N9 | VSS | N9 | VSS | M13 | VSS | 361 | VSS | | | | | |
| 362 | | | | | N10 | VSS | N10 | VSS | N10 | VSS | M14 | VSS | 362 | VSS | | | | | |
| 363 | | | | | N11 | VSS | N11 | VSS | N11 | VSS | M15 | VSS | 363 | VSS | | | | | |
| 364 | | | | | N12 | VSS | N12 | VSS | N12 | VSS | N10 | VSS | 364 | VSS | | | | | |
| 365 | | | | | N14 | VSS | N14 | VSS | N14 | VSS | N11 | VSS | 365 | VSS | | | | | |
| 366 | | | | | P8 | VSS | P8 | VSS | P8 | VSS | N12 | VSS | 366 | VSS | | | | | |
| 367 | | | | | P13 | VSS | P13 | VSS | P13 | VSS | N13 | VSS | 367 | VSS | | | | | |
| 368 | | | | | | | | | | | P9 | VSS | 368 | VSS | | | | | |
| 369 | | | | | | | | | | | P11 | VSS | 369 | VSS | | | | | |
| 370 | | | | | | | | | | | P12 | VSS | 370 | VSS | | | | | |
| 371 | | | | | | | | | | | P14 | VSS | 371 | VSS | | | | | |
| 372 | | | | | | | | | | | R8 | VSS | 372 | VSS | | | | | |
| 373 | | | | | | | | | | | R11 | VSS | 373 | VSS | | | | | |
| 374 | | | | | | | | | | | R12 | VSS | 374 | VSS | | | | | |
| 375 | | | | | | | | | | | R15 | VSS | 375 | VSS | | | | | |
| 376 | | | | | | | | | | | T4 | VSS | 376 | VSS | | | | | |
| 377 | | | | | | | | | | | T5 | VSS | 377 | VSS | | | | | |
| 378 | | | | | | | | | | | U5 | VSS | 378 | VSS | | | | | |
| 379 | | | | | | | | | | | V6 | VSS | 379 | VSS | | | | | |
| 380 | | | | | | | | | | | V14 | VSS | 380 | VSS | | | | | |
| 381 | | | | | | | | | | | AA22 | VSS | 381 | VSS | | | | | |
| 382 | | | | | | | | | | | AB21 | VSS | 382 | VSS | | | | | |
| 383 | | | | | | | | | | | AB22 | VSS | 383 | VSS | | | | | |
| 384 | | | M7 | E0VCC | T10 | E0VCC | T10 | E0VCC | T10 | E0VCC | D9 | E1VCC | 384 | E1VCC | | | | | |
| 385 | | | M10 | E0VCC | M16 | E0VCC | M16 | E0VCC | M16 | E0VCC | D11 | E1VCC | 385 | E1VCC | | | | | |
| 386 | | | C7 | E1VCC | H16 | E1VCC | H16 | E1VCC | H16 | E1VCC | E9 | E1VCC | 386 | E1VCC | | | | | |
| 387 | | | C10 | E1VCC | J5 | E1VCC | J5 | E1VCC | J5 | E1VCC | E11 | E1VCC | 387 | E1VCC | | | | | |
| 388 | | | | | E12 | E1VCC | E12 | E1VCC | E12 | E1VCC | F18 | E1VCC | 388 | E1VCC | | | | | |

## Table 2.1 Pin List (8/9)

| Ref. Nr. | P1M-C Pkg 144 0.4mm pin pitch | | | P1M-C/P1H-C Pkg 156 0.8mm ball pitch | | P1M-C Pkg 292 0.8mm ball pitch | | P1H-C (4MB) Pkg 292 | | P1H-C (8MB) Pkg 292 | | P1H-CE Pkg 404 | | Superset Ref.Nr. | Port name | Port buffer settings Port Input Type | Port Input Type | Port Input Type | Port Input Type | Port Output Type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 389 | | | | | | E8 | E1VCC | E8 | E1VCC | E8 | E1VCC | F19 | E1VCC | 389 | E1VCC | | | | | |
| 390 | | | | | | | | | | | | J4 | E1VCC | 390 | E1VCC | | | | | |
| 391 | | | | | | | | | | | | J5 | E1VCC | 391 | E1VCC | | | | | |
| 392 | | | | | | | | | | | | J18 | E1VCC | 392 | E1VCC | | | | | |
| 393 | | | | | | | | | | | | T18 | E0VCC | 393 | E0VCC | | | | | |
| 394 | | | | | | | | | | | | U18 | E0VCC | 394 | E0VCC | | | | | |
| 395 | | | | | | | | | | | | V13 | E0VCC | 395 | E0VCC | | | | | |
| 396 | | M8 | E0VSS | | | T11 | E0VSS | T11 | E0VSS | T11 | E0VSS | D10 | E1VSS | 396 | E1VSS | | | | | |
| 397 | | M9 | E0VSS | | | N16 | E0VSS | N16 | E0VSS | N16 | E0VSS | D12 | E1VSS | 397 | E1VSS | | | | | |
| 398 | | C6 | E1VSS | | | G16 | E1VSS | G16 | E1VSS | G16 | E1VSS | D18 | E1VSS | 398 | E1VSS | | | | | |
| 399 | | C9 | E1VSS | | | H5 | E1VSS | H5 | E1VSS | H5 | E1VSS | E10 | E1VSS | 399 | E1VSS | | | | | |
| 400 | | | | | | E13 | E1VSS | E13 | E1VSS | E13 | E1VSS | E12 | E1VSS | 400 | E1VSS | | | | | |
| 401 | | | | | | E7 | E1VSS | E7 | E1VSS | E7 | E1VSS | E18 | E1VSS | 401 | E1VSS | | | | | |
| 402 | | | | | | | | | | | | E19 | E1VSS | 402 | E1VSS | | | | | |
| 403 | | | | | | | | | | | | H4 | E1VSS | 403 | E1VSS | | | | | |
| 404 | | | | | | | | | | | | H5 | E1VSS | 404 | E1VSS | | | | | |
| 405 | | | | | | | | | | | | H18 | E1VSS | 405 | E1VSS | | | | | |
| 406 | | | | | | | | | | | | V12 | E0VSS | 406 | E0VSS | | | | | |
| 407 | | | | | | | | | | | | V16 | E0VSS | 407 | E0VSS | | | | | |
| 408 | | | | | | | | | | | | V17 | E0VSS | 408 | E0VSS | | | | | |
| 409 | | | | | | | | | | | | V18 | E0VSS | 409 | E0VSS | | | | | |
| 410 | | | | | | | | | | | | W1 | A1VCC | 410 | A1VCC | | | | | |
| 411 | | | | | | | | | | | | W2 | A1VCC | 411 | A1VCC | | | | | |
| 412 | | | | | | | | | | | | U3 | A1VCC | 412 | A1VCC | | | | | |
| 413 | | | | | | | | | | | | V3 | A1VCC | 413 | A1VCC | | | | | |
| 414 | | | | | | | | | | | | U4 | A1VCC | 414 | A1VCC | | | | | |
| 415 | | | | | | W1 | A1VSS | W1 | A1VSS | W1 | A1VSS | AA1 | A1VSS | 415 | A1VSS | | | | | |
| 416 | | | | | | W2 | A1VSS | W2 | A1VSS | W2 | A1VSS | AA2 | A1VSS | 416 | A1VSS | | | | | |
| 417 | | | | | | Y1 | A1VSS | Y1 | A1VSS | Y1 | A1VSS | W3 | A1VSS | 417 | A1VSS | | | | | |
| 418 | | | | | | | | | | | | V4 | A1VSS | 418 | A1VSS | | | | | |
| 419 | | | | | | | | | | | | Y4 | A0VCC | 419 | A0VCC | | | | | |
| 420 | | | | | | | | | | | | AA4 | A0VCC | 420 | A0VCC | | | | | |
| 421 | | | | | | | | | | | | AB4 | A0VCC | 421 | A0VCC | | | | | |
| 422 | | P1 | A0VSS | | | R5 | A0VSS | R5 | A0VSS | R5 | A0VSS | AB1 | A0VSS | 422 | A0VSS | | | | | |
| 423 | | | | | | T5 | A0VSS | T5 | A0VSS | T5 | A0VSS | AB2 | A0VSS | 423 | A0VSS | | | | | |
| 424 | | | | | | T6 | A0VSS | T6 | A0VSS | T6 | A0VSS | Y3 | A0VSS | 424 | A0VSS | | | | | |
| 425 | | | | | | U4 | A0VSS | U4 | A0VSS | U4 | A0VSS | W4 | A0VSS | 425 | A0VSS | | | | | |
| 426 | | | | | | Y2 | A0VSS | Y2 | A0VSS | Y2 | A0VSS | V5 | A0VSS | 426 | A0VSS | | | | | |
| 427 | | | | | | | | | | | | L4 | VSS | 427 | VSS | | | | | |
| 428 | | | | | | | | | | | | L5 | VSS | 428 | VSS | | | | | |
| 429 | | | | | | | | | | | | V8 | VSS | 429 | VSS | | | | | |
| 430 | | | | | | | | | | | | W8 | VSS | 430 | VSS | | | | | |
| 431 | | | | | | | | | | | | M4 | ERAM VDD | 431 | ERAM VDD | | | | | |
| 432 | | | | | | | | | | | | M5 | ERAMV DD | 432 | ERAM VDD | | | | | |
| 433 | | | | | | | | | | | | V9 | ERAM VDD | 433 | ERAM VDD | | | | | |
| 434 | | | | | | | | | | | | W9 | ERAM VDD | 434 | ERAM VDD | | | | | |
| 435 | | | | | | | | | | | | N4 | ERAM1 VSS | 435 | ERAM1 VSS | | | | | |
| 436 | | | | | | | | | | | | N5 | ERAM1 VSS | 436 | ERAM1 VSS | | | | | |
| 437 | | | | | | | | | | | | V10 | ERAM0 VSS | 437 | ERAM0 VSS | | | | | |
| 438 | | | | | | | | | | | | W10 | ERAM0 VSS | 438 | ERAM0 VSS | | | | | |
| 439 | | | | | | | | | | | | P4 | ERAM1 VCC | 439 | ERAM1 VCC | | | | | |
| 440 | | | | | | | | | | | | P5 | ERAM1 VCC | 440 | ERAM1 VCC | | | | | |
| 441 | | | | | | | | | | | | V11 | ERAM0 VCC | 441 | ERAM0 VCC | | | | | |

## Table 2.1 Pin List (9/9)

| Ref. Nr. | P1M-C Pkg 144 0.4mm pin pitch | P1M-C/P1H-C Pkg 156 0.8mm ball pitch | P1M-C Pkg 292 0.8mm ball pitch | P1H-C (4MB) Pkg 292 | P1H-C (8MB) Pkg 292 | P1H-CE Pkg 404 | | Superset Ref.Nr. | Port name | Port buffer settings: Port Input Type | Port Input Type | Port Input Type | Port Input Type | Port Output Type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 442 | | | | | | W11 | ERAM0 VCC | 442 | ERAM0 VCC | | | | | |
| 443 | | | | | | J20 | DVSS | 443 | DVSS | | | | | |
| 444 | | | | | | J21 | DVSS | 444 | DVSS | | | | | |
| 445 | | | | | | J22 | DVSS | 445 | DVSS | | | | | |
| 446 | | | | | | K20 | DVSS | 446 | DVSS | | | | | |
| 447 | | | | | | L20 | DVSS | 447 | DVSS | | | | | |
| 448 | | | | | | L21 | DVSS | 448 | DVSS | | | | | |
| 449 | | | | | | L22 | DVSS | 449 | DVSS | | | | | |
| 450 | | | | | | M19 | DVSS | 450 | DVSS | | | | | |
| 451 | | | | | | M20 | DVSS | 451 | DVSS | | | | | |
| 452 | | | | | | N20 | DVSS | 452 | DVSS | | | | | |
| 453 | | | | | | N21 | DVSS | 453 | DVSS | | | | | |
| 454 | | | | | | N22 | DVSS | 454 | DVSS | | | | | |
| 455 | | | | | | P20 | DVSS | 455 | DVSS | | | | | |
| 456 | | | | | | R20 | DVSS | 456 | DVSS | | | | | |
| 457 | | | | | | R21 | DVSS | 457 | DVSS | | | | | |
| 458 | | | | | | R22 | DVSS | 458 | DVSS | | | | | |

## 2.2.2 Pin Status

Table 2.2 Pin Status (1/3)

| Pin Function | | | Pin Status | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pin Name | | RESETZ pin = L | RESETZ pin = H | | | | Reset request | |
| | | | | Internal Reset | | | | | |
| Pin Category | | I/F Mode | Before I/F mode is defined | Before I/F mode is defined | Field BIST (TRSTZ = L)*13 | After I/F mode is defined | RUN | @TRSTZ = H & EVA_MODE0.OCD_MD = 1 | @TRSTZ = L or EVA_MODE0.OCD_MD = 0 & TRST = H*11 |
| CLOCK | X1 | — | I | I | I | I | I | I | I |
| | X2 | — | O | O | O | O | O | O | O |
| SYSTEM | RESETZ | — | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) |
| | FLMD0 | — | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) |
| | FLMD1 | — | I *7 | I *7 | I *7 | I *7 | I *7 | I *7 | I *7 |
| | MODE0 | — | I *7 | I *7 | I *7 | I *7 | I *7 | I *7 | I *7 |
| | MODE1 | — | I *7 | I *7 | I *7 | I *7 | I *7 | I *7 | I *7 |
| GPIO | Pn_m | — | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z |
| GPIO | JPn_m | — | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z |
| SAR A/D | ADCAnIm | — | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z |
| | AnVREFH | — | I | I | I | I | I | I | I |
| Other Func. | RESETOUTZ (P0_10) | — | O (Low) | O (Low) | O (Low) | O (Low) | O (Low) | *1 | O (Low) |
| | CVMOUTZ | — | O *2 | O *2 | O (High)*2, *5 | O *2 | O *2, | O *2 | O *2 |
| | ERROROUTZ | — | Hi-Z | O(Low) *3, *6 | O(Low) *3, *6 | O(Low) *3, *6 | O *3 | Hi-Z *14 | Hi-Z |

Table 2.2 Pin Status (2/3)

| Pin Function | | | Pin Status | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pin Name | | RESETZ pin = L | RESETZ pin = H | | | | Reset request | |
| | | | | Internal Reset | | | | | |
| Pin Category | | I/F Mode | Before I/F mode is defined | Before I/F mode is defined | Field BIST (TRSTZ = L)*13 | After I/F mode is defined | RUN | @TRSTZ = H & EVA_MODE0.OCD_MD = 1 | @TRSTZ = L or EVA_MODE0.OCD_MD = 0 & TRST = H*11 |
| JTAG | TDI/LPDI | NEXUS (TDI) | Hi-Z | Hi-Z | Hi-Z | I (Pull-up) | I (Pull-up) | I (Pull-up) | Hi-Z |
| | | LPD-4pin (LPDI) | Hi-Z | Hi-Z | Hi-Z | I (Pull-up) | I (Pull-up) | I (Pull-up) | Hi-Z |
| | | Boot mode (JP0_0) | Hi-Z | — | — | Hi-Z | Hi-Z | —*16 | Hi-Z |
| | | BSCAN (TDI) | Hi-Z | — | — | I (Pull-up) | I (Pull-up) | —*10 | Hi-Z |
| | TDO/LPDO | NEXUS (TDO) | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z |
| | | LPD-4pin (LPDO) | Hi-Z | Hi-Z | Hi-Z | O | O | O | Hi-Z |
| | | Boot mode (JP0_1) | Hi-Z | — | — | Hi-Z | Hi-Z | —*16 | Hi-Z |
| | | BSCAN (TDO) | Hi-Z | — | — | Hi-Z | Hi-Z | —*10 | Hi-Z |
| | TCK/ LPDCLKI | NEXUS (TCK) | Hi-Z | Hi-Z | Hi-Z | I (Pull-up) | I (Pull-up) | I (Pull-up) | Hi-Z |
| | | LPD-4pin (LPDCLKI) | Hi-Z | Hi-Z | Hi-Z | I (Pull-up) | I (Pull-up) | I (Pull-up) | Hi-Z |
| | | Boot mode (JP0_2) | Hi-Z | — | — | Hi-Z | Hi-Z | —*16 | Hi-Z |
| | | BSCAN (TCK) | Hi-Z | — | — | I (Pull-up) | I (Pull-up) | —*10 | Hi-Z |
| | TMS | NEXUS (TMS) | Hi-Z | Hi-Z | Hi-Z | I (Pull-up) | I (Pull-up) | I (Pull-up) | Hi-Z |
| | | LPD-4pin (No function) | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z |
| | | Boot mode (JP0_3) | Hi-Z | — | — | Hi-Z | Hi-Z | —*16 | Hi-Z |
| | | BSCAN (TMS) | Hi-Z | — | — | I (Pull-up) | I (Pull-up) | —*10 | Hi-Z |
| | TRSTZ/ LPDRSTZ | NEXUS (TRSTZ) (*9) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) |
| | | LPD-4pin (LPDRSTZ) (*9) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) |
| | | Boot mode (TRSTZ) (*9) | I (Pull-down) | — | — | I (Pull-down) | I (Pull-down) | —*16 | I (Pull-down) |
| | | BSCAN (TRSTZ) | I (Pull-down) | — | — | I (Pull-down) | I (Pull-down) | —*10 | I (Pull-down) |
| | RDY | NEXUS (RDY) | Hi-Z | Hi-Z | Hi-Z | Hi-Z*15 | O | O | Hi-Z |
| | | LPD-4pin (LPDCLKOUT) | Hi-Z | Hi-Z | Hi-Z | O | O | O | Hi-Z |
| | | Boot mode (JP0_5) | Hi-Z | — | — | Hi-Z | Hi-Z | —*16 | Hi-Z |
| | | BSCAN (No function) | Hi-Z | — | — | Hi-Z | Hi-Z | —*10 | Hi-Z |
| AUDR (*12, only for P1H-CE) | AUDRSTZ | — | Hi-Z | Hi-Z | Hi-Z | Hi-Z | I *8 | Hi-Z | Hi-Z |
| | AUDRCK | — | Hi-Z | Hi-Z | Hi-Z | Hi-Z | I *8 | Hi-Z | Hi-Z |
| | AUDSYNC | — | Hi-Z | Hi-Z | Hi-Z | Hi-Z | I *8 | Hi-Z | Hi-Z |
| | AUDATA0 | — | Hi-Z | Hi-Z | Hi-Z | Hi-Z | I *8 | Hi-Z | Hi-Z |
| | AUDATA1 | — | Hi-Z | Hi-Z | Hi-Z | Hi-Z | I *8 | Hi-Z | Hi-Z |
| | AUDATA2 | — | Hi-Z | Hi-Z | Hi-Z | Hi-Z | I *8 | Hi-Z | Hi-Z |
| | AUDATA3 | — | Hi-Z | Hi-Z | Hi-Z | Hi-Z | I *8 | Hi-Z | Hi-Z |

**Table 2.2 Pin Status (3/3)**

| Pin Function | | | Pin Status | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pin Name | | RESETZ pin = L | RESETZ pin = H | | | | Reset request | |
| | | | | Internal Reset | | | | | |
| Pin Category | | I/F Mode | Before I/F mode is defined | Before I/F mode is defined | Field BIST (TRSTZ = L)*13 | After I/F mode is defined | RUN | @TRSTZ = H & EVA_MODE0.OCD_MD = 1 | @TRSTZ = L or EVA_MODE0.OCD_MD = 0 & TRST = H*11 |
| AURORA (*12, only for P1H-CE) | CICREFP | — | I | I | I | I | I | I | I |
| | CICREFN | — | I | I | I | I | I | I | I |
| | TODP0 | — | O | O | O | O | O | O | O |
| | TODN0 | — | O | O | O | O | O | O | O |
| | TODP1 | — | O | O | O | O | O | O | O |
| | TODN1 | — | O | O | O | O | O | O | O |
| | AuroraReset1 | — | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) |
| | AuroraReset1 PD | — | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) |
| | AuroraReset2 | — | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) |
| ERAM (*12, only for P1H-CE) | ERAMReset1 PD | — | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) |
| | ERAMReset2 | — | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) | I (Pull-down) |
| Other Debug (*12, only for P1H-CE) | EVTIZ | — | I (Pull-up) | I (Pull-up) | I (Pull-up) | I (Pull-up) | I (Pull-up) | I (Pull-up) | I (Pull-up) |
| | EVTOZ | — | O | O | O | O | O | O | O |
| | MSYNZ | — | I (Pull-up) | I (Pull-up) | I (Pull-up) | I (Pull-up) | I (Pull-up) | I (Pull-up) | I (Pull-up) |

Note 1. RESETOUTZ pin drive out low level during and after reset. Until being disabled by port register settings, this pin keeps driveing out low level.

Note 2. CVMOUTZ pin level depends on core voltage status. And this pin has read back function to check pin level.

Note 3. ERROROUTZ pin is reseted to TriState level while CVM Reset. This pin has read back function to check pin level.

Note 4. FLMD0, FLMD1, MODE0, MODE1 signals are latched at RESETZ rising timing.

Note 5. During Field BIST execution, CVMOUTZ pin status is depends on CVMFBISTME bit in CVMDE register. For details, see **Section 10.3.5, CVMDE — CVM detection enable register** in **Section 10, Core Voltage Monitor (CVM)**.

Note 6. ERROROUTZ pin is Low-clamp until Field BIST execution is done.

Note 7. FLMD1, MODE0, MODE1 inputs are masked by FLMD0 value.

Note 8. Hi-Z when AUDR is disabled. Regarding of a setting to enable AUDR, see *Section 33.12.1 OPBT0 and RH850/P1x-C User's Manual: Hardware (Security)*.

Note 9. JP0_4 can be used only as input pin and its pin level can only be read by JPPR0 register in Port mode.

Note 10. "EVA_MODE0.OCD_MD = 1" cannot be given during Boundary Scan mode.

Note 11. SCDS is initialized when EVA_MODE0.OCD_MD = 0 (not reset mask) & RESETZ = L. Therefore, JTAG pins are also initialized like "RESETZ pin = L" status.

Note 12. Debug pins run only on the emulation device.

Note 13. Field BIST status will be skipped if it set TRSTZ = H.

Note 14. ERROROUTZ = Low during "RESETZ = L & TRSTZ = H & EVA_MODE0.OCD_MD = 1" at "Application Reset 1" and "System Reset 2".

Note 15. RDYZ = OUT status during TRSTZ = H.

Note 16. "EVA_MODE0.OCD_MD = 1" cannot be given during bootmode mode (Serial Programming Mode).

## 2.2.3 Pin Function assignments

Table 2.3 Pin Function assignments (1/6)

| Superset Ref.No. | Port name | 1st Alternative | | | | | | | | | | | | 2nd Alternative | | | | | | | | | | | | 3rd Alternative | | | | | | | | | | | | 4th Alternative | | | | | | | | | | | | Special Function | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | In | | | | | | | | Out | | | | In | | | | | | | | Out | | | | In | | | | | | | | Out | | | | In | | | | | | | | Out | | | | | | | |
| | | Assignment | | | | | | | | Assignment | | | | Assignment | | | | | | | | Assignment | | | | Assignment | | | | | | | | Assignment | | | | Assignment | | | | | | | | Assignment | | | | | | | |
| | | Function name | PIM-C (QFP-144) | PIM-C/PIH-C (BGA-156) | PIM-C (BGA-292) | PIH-C (MMB) | PIH-C (BMB) | PIH-CE | Function name | PIM-C (QFP-144) | PIM-C/PIH-C (BGA-156) | PIM-C (BGA-292) | PIH-C (MMB) | PIH-C (BMB) | PIH-CE | Function name | PIM-C (QFP-144) | PIM-C/PIH-C (BGA-156) | PIM-C (BGA-292) | PIH-C (MMB) | PIH-C (BMB) | PIH-CE | Function name | PIM-C (QFP-144) | PIM-C/PIH-C (BGA-156) | PIM-C (BGA-292) | PIH-C (MMB) | PIH-C (BMB) | PIH-CE | Function name | PIM-C (QFP-144) | PIM-C/PIH-C (BGA-156) | PIM-C (BGA-292) | PIH-C (MMB) | PIH-C (BMB) | PIH-CE | Function name | PIM-C (QFP-144) | PIM-C/PIH-C (BGA-156) | PIM-C (BGA-292) | PIH-C (MMB) | PIH-C (BMB) | PIH-CE | Function name | PIM-C (QFP-144) | PIM-C/PIH-C (BGA-156) | PIM-C (BGA-292) | PIH-C (MMB) | PIH-C (BMB) | PIH-CE |
| 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | P3_14 | GTM01 | √ | √ | √ | √ | √ | √ | GTMAT0O1 | √ | √ | √ | √ | √ | √ | RLIN30RX | √ | √ | √ | √ | √ | √ | ETH0TXEN | √ | √ | √ | √ | √ | √ | MCAN0RX | √ | √ | √ | √ | √ | √ | | | | | | | FLX0RXDA/ INTP10 | √ | √ | √ | √ | √ | √ | | | | | | |
| 3 | P4_0 | GTM10 | √ | √ | √ | √ | √ | √ | GTMAT0O0 | √ | √ | √ | √ | √ | √ | ETH0RXER | √ | √ | √ | √ | √ | √ | | | | | | | | | | | | | MCAN0TX | √ | √ | √ | √ | √ | √ | FLX0RXDB | √ | √ | √ | √ | √ | √ | RLIN30TX | √ | √ | √ | √ | √ | √ | | | | | | |
| 4 | P4_1 | GTM02 | √ | √ | √ | √ | √ | √ | GTMAT0O2 | √ | √ | √ | √ | √ | √ | ETH0TXCLK | √ | √ | √ | √ | √ | √ | | | | | | | CSIH1SSIZ | √ | √ | √ | √ | √ | √ | CSIH1CSS0 | √ | √ | √ | √ | √ | √ | FLX0STPWT | √ | √ | √ | √ | √ | √ | | | | | | |
| 5 | P4_2 (MODE0) | GTM11 | √ | √ | √ | √ | √ | √ | GTMAT0O1 | √ | √ | √ | √ | √ | √ | ETH0RXCLK | √ | √ | √ | √ | √ | √ | | | | | | | CSIH1DCS | √ | √ | √ | √ | √ | √ | CSIH1SO0 | √ | √ | √ | √ | √ | √ | | | | | | | FLX0TXENA | √ | √ | √ | √ | √ | √ | MODE0 | √ | √ | √ | √ | √ | √ |
| 6 | VDD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | P4_3 (MODE1) | GTM03 | √ | √ | √ | √ | √ | √ | GTMAT0O0N | √ | √ | √ | √ | √ | √ | ETH0RXD0 | √ | √ | √ | √ | √ | √ | | | | | | | CSIH1SCI0 | √ | √ | √ | √ | √ | √ | CSIH1SCO0 | √ | √ | √ | √ | √ | √ | | | | | | | FLX0TXDB | √ | √ | √ | √ | √ | √ | MODE1 | √ | √ | √ | √ | √ | √ |
| 8 | P4_4 | GTM06 | √ | √ | √ | √ | √ | √ | GTMAT0O2 | √ | √ | √ | √ | √ | √ | ETH0RXD1 | √ | √ | √ | √ | √ | √ | RLIN30TX | √ | √ | √ | √ | √ | √ | CSIH1SI0 | √ | √ | √ | √ | √ | √ | | | | | | | | | | | | | FLX0TXDA | √ | √ | √ | √ | √ | √ | | | | | | |
| 9 | P4_5 (FLMD1) | GTM14 | √ | √ | √ | √ | √ | √ | GTMAT0O0 | √ | √ | √ | √ | √ | √ | ETH0RXD2 | √ | √ | √ | √ | √ | √ | | | | | | | MTTCAN0RX | √ | √ | √ | √ | √ | √ | | | | | | | RLIN30RX | √ | √ | √ | √ | √ | √ | FLX0TXENB | √ | √ | √ | √ | √ | √ | FLMD1 | √ | √ | √ | √ | √ | √ |
| 10 | P4_6 | GTM15 | √ | √ | √ | √ | √ | √ | GTMAT0O5 | √ | √ | √ | √ | √ | √ | ETH0RXD3 | √ | √ | √ | √ | √ | √ | RLIN30TX | √ | √ | √ | √ | √ | √ | | | | | | | MTTCAN0TX | √ | √ | √ | √ | √ | √ | | | | | | | | | | | | | | | | | | |
| 11 | P4_7 | GTM11 | √ | √ | √ | √ | √ | √ | GTMAT0O5 | √ | √ | √ | √ | √ | √ | ETH0RXDV | √ | √ | √ | √ | √ | √ | | | | | | | INTP0 | √ | √ | √ | √ | √ | √ | EXTCLK0O | √ | √ | √ | √ | √ | √ | SENT0RX | √ | √ | √ | √ | √ | √ | BHPDGRCLK0 | √ | √ | √ | √ | √ | √ | | | | | | |
| 12 | E1VSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | E1 VCC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | P4_8 | GTM10 | √ | √ | √ | √ | √ | √ | GTMAT0O0N | √ | √ | √ | √ | √ | √ | | | | | | | | ETH0WOL | √ | √ | √ | √ | √ | √ | RLIN30RX | √ | √ | √ | √ | √ | √ | | | | | | | SENT1RX | √ | √ | √ | √ | √ | √ | BHPDGRCLK1 | √ | √ | √ | √ | √ | √ | | | | | | |
| 15 | P4_9 | GTM14 | √ | √ | √ | √ | √ | √ | GTMAT0O4 | √ | √ | √ | √ | √ | √ | | | | | | | | EXTCLK1O | √ | √ | √ | √ | √ | √ | CSIH2SSIZ | √ | √ | √ | √ | √ | √ | CSIH2CSS0 | √ | √ | √ | √ | √ | √ | | | | | | | RLIN30TX | √ | √ | √ | √ | √ | √ | | | | | | |
| 16 | P4_10 | GTM03 | √ | √ | √ | √ | √ | √ | GTMAT1O1N | √ | √ | √ | √ | √ | √ | MCAN1RX | √ | √ | √ | √ | √ | √ | | | | | | | CSIH2DCS | √ | √ | √ | √ | √ | √ | CSIH2SO2 | √ | √ | √ | √ | √ | √ | | | | | | | CSIH0CSS3 | √ | √ | √ | √ | √ | √ | | | | | | |
| 17 | P4_11 | GTM02 | √ | √ | √ | √ | √ | √ | GTMAT1O2N | √ | √ | √ | √ | √ | √ | | | | | | | | MCAN1TX | √ | √ | √ | √ | √ | √ | CSIH2SCI2 | √ | √ | √ | √ | √ | √ | CSIH2SCO2 | √ | √ | √ | √ | √ | √ | | | | | | | CSIH1CSS4 | √ | √ | √ | √ | √ | √ | | | | | | |
| 18 | P4_12 | GTM01 | √ | √ | √ | √ | √ | √ | GTMAT1O3N | √ | √ | √ | √ | √ | √ | INTP1 | √ | √ | √ | √ | √ | √ | FLXNTUOUT | √ | √ | √ | √ | √ | √ | CSIH2SI2 | √ | √ | √ | √ | √ | √ | | | | | | | ADC1TRG | √ | √ | √ | √ | √ | √ | CSIH1CSS5 | √ | √ | √ | √ | √ | √ | | | | | | |
| 19 | VDD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 | VSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 21 | VCC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 22 | P4_13 | GTM17 | √ | √ | √ | √ | √ | √ | GTMAT1O0 | √ | √ | √ | √ | √ | √ | | | | | | | | CSIH0CSS5 | √ | √ | √ | √ | √ | √ | SENT2RX | √ | √ | √ | √ | √ | √ | | | | | | | ESO1Z | √ | √ | √ | √ | √ | √ | CSIH2CSS7 | √ | √ | √ | √ | √ | √ | | | | | | |
| 23 | P4_14 | GTM17 | √ | | √ | √ | √ | √ | GTMAT1O0N | √ | | √ | √ | √ | √ | | | | | | | | CSIH0CSS6 | √ | | √ | √ | √ | √ | SENT3RX | √ | | √ | √ | √ | √ | | | | | | | | | | | | | EXTCLK1O | √ | | √ | √ | √ | √ | | | | | | |
| 24 | P6_0 | GTM16 | √ | | √ | √ | √ | √ | GTMAT1O1 | √ | | √ | √ | √ | √ | | | | | | | | | | | | | | | SENT4RX | √ | | √ | √ | √ | √ | | | | | | | | | | | | | CSIH1CSS6 | √ | | √ | √ | √ | √ | ICUMGPIO0 | √ | | √ | √ | √ | √ |
| 25 | P6_1 | GTM16 | √ | | √ | √ | √ | √ | GTMAT1O1N | √ | | √ | √ | √ | √ | | | | | | | | | | | | | | | SENT5RX | √ | | √ | √ | √ | √ | | | | | | | | | | | | | CSIH0CSS7 | √ | | √ | √ | √ | √ | ICUMGPIO1 | √ | | √ | √ | √ | √ |
| 26 | ERAMRESPDZ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 27 | ERAMRES2Z | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 28 | ADC1I0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 29 | ADC1I1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 30 | ADC1I2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 31 | ADC1I3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 32 | ADC1I4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 33 | ADC1I5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 34 | ADC1I6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 35 | ADC1I7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 36 | ADC1I8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 37 | ADC1I9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 38 | ADC1I10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 39 | ADC1I11 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ | √ | √ | √ |
| 40 | ADC1I12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ |
| 41 | ADC1I13 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ |
| 42 | ADC1I14 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ | √ | √ |

## Table 2.3 Pin Function assignments (2/6)

| Superset Ref.No. | Port name | 1st Alternative | | | | | | | | | | | | 2nd Alternative | | | | | | | | | | | | 3rd Alternative | | | | | | | | | | | | 4th Alternative | | | | | | | | | | | | Special Function | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | In | | | | | | | | Out | | | | In | | | | | | | | Out | | | | In | | | | | | | | Out | | | | In | | | | | | | | Out | | | | Special Function | | | | |
| | | Assignment | | | | | | | | Assignment | | | | Assignment | | | | | | | | Assignment | | | | Assignment | | | | | | | | Assignment | | | | Assignment | | | | | | | | Assignment | | | | Assignment | | | | |
| | | Function name | P1M-C (QFP-144) | P1M-CP1H-C (BGA-156) | P1M-C (BGA-292) | P1H-C (MB) | P1H-C (SMB) | P1H-CE

## Table 2.3 Pin Function assignments (3/6)

| Superset Ref.No. | Port name | 1st Alternative | | | | | | | | | | | | | | | | 2nd Alternative | | | | | | | | | | | | | | | | 3rd Alternative | | | | | | | | | | | | | | | | 4th Alternative | | | | | | | | | | | | | | | | Special Function | | | | |
|---|---|---

## Table 2.3 Pin Function assignments (4/6)

| Superset Ref.No. | Port name | 1st Alternative | | | | | | | | 2nd Alternative | | | | | | | | 3rd Alternative | | | | | | | | 4th Alternative | | | | | | | | Special Function | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | In | | | | | | Out | | | | | | In | | | | | | Out | | | | | | In | | | | | | Out | | | | | | | |
| | | Assignment | | | | | | Assignment | | | | | | Assignment | | | | | | Assignment | | | | | | Assignment | | | | | | Assignment | | | | | | | |
| | | Function name | P1M-C (QFP-144) | P1M-CP1H-C (80A-156) | P1M-C (BGA-292) | P1H-C (4MB) | P1H-C (8MB) | P1H-CE | Function name | P1M-C (QFP-144) | P1M-CP1H-C (80A-156) | P1M-C (BGA-292) | P1H-C (4MB) | P1H-C (8MB) | P1H-CE | Function name | P1M-C (QFP-144) | P1M-CP1H-C (80A-156) | P1M-C (BGA-292) | P1H-C (4MB) | P1H-C (8MB) | P1H-CE | Function name | P1M-C (QFP-144) | P1M-CP1H-C (80A-156) | P1M-C (BGA-292) | P1H-C (4MB) | P1H-C (8MB) | P1H-CE | Function name | P1M-C (QFP-144) | P1M-CP1H-C (80A-156) | P1M-C (BGA-292) | P1H-C (4MB) | P1H-C (8MB) | P1H-CE | Function name | P1M-C (QFP-144) | P1M-CP1H-C (80A-156) | P1M-C (BGA-292) | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
| 133 | P0_8 | GTM0I5 | √ | √ | √ | √ | √ | √ | GTMAT1O3 | √ | √ | √ | √ | √ | √ | MCAN1TX | √ | √ | √ | √ | √ | √ | CSIH0SCI2 | √ | √ | √ | √ | √ | √ | CSIH0SCO2 | √ | √ | √ | √ | √ | √ | CSIH1SCI2 | √ | √ | √ | √ | √ | √ | CSIH1SCO2 | √ | √ | √ | √ | √ | √ | | | | | | | |
| 134 | P0_9 | GTM1I4 | √ | √ | √ | √ | √ | √ | GTMAT1O0N | √ | √ | √ | √ | √ | √ | EXTCLK1O | √ | √ | √ | √ | √ | √ | CSIH0DCS | √ | √ | √ | √ | √ | √ | CSIH0SO2 | √ | √ | √ | √ | √ | √ | CSIH1SI2 | √ | √ | √ | √ | √ | √ | | | | | | | | | | | | | |
| 135 | FLMD0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 136 | CVMOUTZ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 137 | E0VSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 138 | E0 VCC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 139 | P0_10 (RESETOUTZ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RESETOUTZ | √ | √ | √ | √ | √ | √ |
| 140 | VDD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 141 | VSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 142 | P0_13 | GTM1I3 | √ | √ | √ | √ | √ | √ | GTMAT1O1N | √ | √ | √ | √ | √ | √ | CSIH1RYI | √ | √ | √ | √ | √ | √ | CSIH1RYO | √ | √ | √ | √ | √ | √ | SENT4RX | √ | √ | √ | √ | √ | √ | MCAN0RX | √ | √ | √ | √ | √ | √ | | | | | | | | | | | | | |
| 143 | P0_14 | GTM0I2 | √ | √ | √ | √ | √ | √ | GTMAT0O4 | √ | √ | √ | √ | √ | √ | EXTCLK1O | √ | √ | √ | √ | √ | √ | SENT0RX | √ | √ | √ | √ | √ | √ | | | | | | | | | | | | | | | | | | | MCAN0TX | √ | √ | √ | √ | √ | √ | BHPDGREN | √ | √ | √ | √ | √ | √ |
| 144 | MSYNZ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 145 | EMU VCC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 146 | EMU VSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 147 | AURORES2Z | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 148 | AURORES1Z | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 149 | AURORESPDZ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 150 | EMU VDD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 151 | EMU VDD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 152 | VSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | VSS | | | | | | √ |
| 153 | VSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | VSS | | | | | | √ |
| 154 | DVDD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 155 | TODP0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 156 | TODN0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 157 | DVSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 158 | TODP1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 159 | TODN1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 160 | DVDD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 161 | DVSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 162 | DVSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 164 | CICREFP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 165 | CICREFN | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 166 | DVCC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 167 | DVCC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 168 | DVSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 169 | VSS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | VSS | √ | √ | √ | √ | √ | √ |
| 170 | JP0_5 (RDYZ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RDYZ | √ | √ | √ | √ | √ | √ |
| 171 | JP0_3 (TMS) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | TMS | √ | √ | √ | √ | √ | √ |
| 172 | JP0_2 (TCK) | | | | | | | | | | | | | | | | | | | | | | | | | FLSCI3SCKI | √ | √ | √ | √ | √ | √ | | | | | | | | | | | | | | | | | | | TCK | √ | √ | √ | √ | √ | √ |
| 175 | JP0_1 (TDO) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | FLSCI3TXD | √ | √ | √ | √ | √ | √ | TDO | √ | √ | √ | √ | √ | √ |
| 176 | JP0_0 (TDI) | | | | | | | | | | | | | | | | | | | | | | | | | FLSCI3RXD | √ | √ | √ | √ | √ | √ | FLSCI3TXD | √ | √ | √ | √ | √ | √ | | | | | | | | | | | | | | | | | | | TDI | √ | √ | √ | √ | √ | √ |
| 177 | JP0_4 (TRSTZ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | TRSTZ | √ | √ | √ | √ | √ | √ |
| 178 | EVTOZ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 179 | EVTIZ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |
| 182 | AUDATA3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | √ |

## Table 2.3 Pin Function assignments (5/6)

| Superset Ref.No. | Port name | 1st Alternative | | | | | | | | | | | | | | | | | | | | | | 2nd Alternative | | | | | | | | | | | | | | | | | | | | | | 3rd Alternative | | | | | | | | | | | | | | | | | | | | | | 4th Alternative | | | | | | | | | | | | | | | | | | | | | | Special Function | | | | | | | | | |
|---|---|---

## Table 2.3 Pin Function assignments (6/6)

| Superset Ref.No. | Port name | 1st Alternative | | | | | | | | | | | | | | | | 2nd Alternative | | | | | | | | | | | | | | | | 3rd Alternative | | | | | | | | | | | | | | | | 4th Alternative | | | | | | | | | | | | | | | | Special Function | | | | | | | | | |
|---|---|---

## 2.2.4 Pin Function name

Table 2.4 Pin Function Name Definition (1/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 1 | NMI | I | Pin (NMI) | non maskable interrupt |
| 2 | INTP0 | I | Pin (INTP) | maskable external interrupt 0 |
| 3 | INTP1 | I | | maskable external interrupt 1 |
| 4 | INTP2 | I | | maskable external interrupt 2 |
| 5 | INTP3 | I | | maskable external interrupt 3 |
| 6 | INTP4 | I | | maskable external interrupt 4 |
| 7 | INTP5 | I | | maskable external interrupt 5 |
| 8 | INTP6 | I | | maskable external interrupt 6 |
| 9 | INTP7 | I | | maskable external interrupt 7 |
| 10 | INTP8 | I | | maskable external interrupt 8 |
| 11 | INTP9 | I | | maskable external interrupt 9 |
| 12 | INTP10 | I | | maskable external interrupt 10 |
| 13 | INTP11 | I | | maskable external interrupt 11 |
| 14 | EXTCLK0O | O | System Control | Clock controller output |
| 15 | EXTCLK1O | O | | Clock controller output |
| 16 | CSIH0CSS0 | O | CSIH0 | CSIH0 serial peripheral chip select signal 0 |
| 17 | CSIH0CSS1 | O | | CSIH0 serial peripheral chip select signal 1 |
| 18 | CSIH0CSS2 | O | | CSIH0 serial peripheral chip select signal 2 |
| 19 | CSIH0CSS3 | O | | CSIH0 serial peripheral chip select signal 3 |
| 20 | CSIH0CSS4 | O | | CSIH0 serial peripheral chip select signal 4 |
| 21 | CSIH0CSS5 | O | | CSIH0 serial peripheral chip select signal 5 |
| 22 | CSIH0CSS6 | O | | CSIH0 serial peripheral chip select signal 6 |
| 23 | CSIH0CSS7 | O | | CSIH0 serial peripheral chip select signal 7 |
| 24 | CSIH0SSIZ | I | | CSIH0 serial SS function control input signal |
| 25 | CSIH0RYI | I | | CSIH0 ready (1) / busy (0) input signal |
| 26 | CSIH0RYO | O | | CSIH0 ready (1) / busy (0) output signal |
| 27 | CSIH0SCIn | I | | CSIH0 serial clock input signal n = 0 to 2 |
| 28 | CSIH0SCOn | O | | CSIH0 serial clock output signal n = 0 to 2 |
| 29 | CSIH0SIn | I | | CSIH0 serial data input n = 0 to 2 |
| 30 | CSIH0SOn | O | | CSIH0 serial data output n = 0 to 2 |
| 31 | CSIH0DCSn | I | | CSIH0 data consistency check signal n = 0 to 2 |
| 32 | CSIH1CSS0 | O | CSIH1 | CSIH1 serial peripheral chip select signal 0 |
| 33 | CSIH1CSS1 | O | | CSIH1 serial peripheral chip select signal 1 |
| 34 | CSIH1CSS2 | O | | CSIH1 serial peripheral chip select signal 2 |
| 35 | CSIH1CSS3 | O | | CSIH1 serial peripheral chip select signal 3 |
| 36 | CSIH1CSS4 | O | | CSIH1 serial peripheral chip select signal 4 |
| 37 | CSIH1CSS5 | O | | CSIH1 serial peripheral chip select signal 5 |
| 38 | CSIH1CSS6 | O | | CSIH1 serial peripheral chip select signal 6 |
| 39 | CSIH1CSS7 | O | | CSIH1 serial peripheral chip select signal 7 |
| 40 | CSIH1SSIZ | I | | CSIH1 serial SS function control input signal |
| 41 | CSIH1RYI | I | | CSIH1 ready (1) / busy (0) input signal |
| 42 | CSIH1RYO | O | | CSIH1 ready (1) / busy (0) output signal |

Table 2.4 Pin Function Name Definition (2/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 43 | CSIH1SCIn | I | CSIH1 | CSIH1 serial clock input signal n = 0 to 2 |
| 44 | CSIH1SCOn | O | | CSIH1 serial clock output signal n = 0 to 2 |
| 45 | CSIH1SIn | I | | CSIH1 serial data input n = 0 to 2 |
| 46 | CSIH1SOn | O | | CSIH1 serial data output n = 0 to 2 |
| 47 | CSIH1DCSn | I | | CSIH1 data consistency check signal n = 0 to 2 |
| 48 | CSIH2CSS0 | O | CSIH2 | CSIH2 serial peripheral chip select signal 0 |
| 49 | CSIH2CSS1 | O | | CSIH2 serial peripheral chip select signal 1 |
| 50 | CSIH2CSS2 | O | | CSIH2 serial peripheral chip select signal 2 |
| 51 | CSIH2CSS3 | O | | CSIH2 serial peripheral chip select signal 3 |
| 52 | CSIH2CSS4 | O | | CSIH2 serial peripheral chip select signal 4 |
| 53 | CSIH2CSS5 | O | | CSIH2 serial peripheral chip select signal 5 |
| 54 | CSIH2CSS6 | O | | CSIH2 serial peripheral chip select signal 6 |
| 55 | CSIH2CSS7 | O | | CSIH2 serial peripheral chip select signal 7 |
| 56 | CSIH2SSIZ | I | | CSIH2 serial SS function control input signal |
| 57 | CSIH2RYI | I | | CSIH2 ready (1) / busy (0) input signal |
| 58 | CSIH2RYO | O | | CSIH2 ready (1) / busy (0) output signal |
| 59 | CSIH2SCIn | I | | CSIH2 serial clock input signal n = 0 to 2 |
| 60 | CSIH2SCOn | O | | CSIH2 serial clock output signal n = 0 to 2 |
| 61 | CSIH2SIn | I | | CSIH2 serial data input n = 0 to 2 |
| 62 | CSIH2SOn | O | | CSIH2 serial data output n = 0 to 2 |
| 63 | CSIH2DCSn | I | | CSIH2 data consistency check signal n = 0 to 2 |
| 64 | CSIH3CSS0 | O | CSIH3 | CSIH3 serial peripheral chip select signal 0 |
| 65 | CSIH3CSS1 | O | | CSIH3 serial peripheral chip select signal 1 |
| 66 | CSIH3CSS2 | O | | CSIH3 serial peripheral chip select signal 2 |
| 67 | CSIH3CSS3 | O | | CSIH3 serial peripheral chip select signal 3 |
| 68 | CSIH3CSS4 | O | | CSIH3 serial peripheral chip select signal 4 |
| 69 | CSIH3CSS5 | O | | CSIH3 serial peripheral chip select signal 5 |
| 70 | CSIH3CSS6 | O | | CSIH3 serial peripheral chip select signal 6 |
| 71 | CSIH3CSS7 | O | | CSIH3 serial peripheral chip select signal 7 |
| 72 | CSIH3SSIZ | I | | CSIH3 serial SS function control input signal |
| 73 | CSIH3RYI | I | | CSIH3 ready (1) / busy (0) input signal |
| 74 | CSIH3RYO | O | | CSIH3 ready (1) / busy (0) output signal |
| 75 | CSIH3SCIn | I | | CSIH3 serial clock input signal n = 0 to 2 |
| 76 | CSIH3SCOn | O | | CSIH3 serial clock output signal n = 0 to 2 |
| 77 | CSIH3SIn | I | | CSIH3 serial data input n = 0 to 2 |
| 78 | CSIH3SOn | O | | CSIH3 serial data output n = 0 to 2 |
| 79 | CSIH3DCSn | I | | CSIH3 data consistency check signal n = 0 to 2 |
| 80 | RLIN30RX | I | RLIN30 | RLIN30 data input |
| 81 | RLIN30TX | O | | RLIN30 data output |
| 82 | RLIN31RX | I | RLIN31 | RLIN31 data input |
| 83 | RLIN31TX | O | | RLIN31 data output |
| 84 | RLIN32RX | I | RLIN32 | RLIN32 data input |
| 85 | RLIN32TX | O | | RLIN32 data output |

Table 2.4 Pin Function Name Definition (3/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 86 | RLIN33RX | I | RLIN33 | RLIN33 data input |
| 87 | RLIN33TX | O | | RLIN33 data output |
| 88 | HSURT0SDIO0I | I | HSURT0 | HSURT0 data input 0 |
| 89 | HSURT0SDIO0O | O | | HSURT0 data output 0 |
| 90 | HSURT0SDIO1I | I | | HSURT0 data input 1 |
| 91 | HSURT0SDIO1O | O | | HSURT0 data output 1 |
| 92 | HSURT0SDIO2I | I | | HSURT0 data input 2 |
| 93 | HSURT0SDIO2O | O | | HSURT0 data output 2 |
| 94 | HSURT0SDIO3I | I | | HSURT0 data input 3 |
| 95 | HSURT0SDIO3O | O | | HSURT0 data output 3 |
| 96 | HSURT0SCKI | I | | HSURT0 serial clock input |
| 97 | HSURT0SCKO | O | | HSURT0 serial clock output |
| 98 | HSURT0SDIR | O | | HSURT0 direction indication output |
| 99 | HSURT0CSI | I | | HSURT0 chip select input |
| 100 | HSURT0CSO | O | | HSURT0 chip select output |
| 101 | HSURT1SDIO0I | I | HSURT1 | HSURT1 data input 0 |
| 102 | HSURT1SDIO0O | O | | HSURT1 data output 0 |
| 103 | HSURT1SDIO1I | I | | HSURT1 data input 1 |
| 104 | HSURT1SDIO1O | O | | HSURT1 data output 1 |
| 105 | HSURT1SDIO2I | I | | HSURT1 data input 2 |
| 106 | HSURT1SDIO2O | O | | HSURT1 data output 2 |
| 107 | HSURT1SDIO3I | I | | HSURT1 data input 3 |
| 108 | HSURT1SDIO3O | O | | HSURT1 data output 3 |
| 109 | HSURT1SCKI | I | | HSURT1 serial clock input |
| 110 | HSURT1SCKO | O | | HSURT1 serial clock output |
| 111 | HSURT1SDIR | O | | HSURT1 direction indication output |
| 112 | HSURT1CSI | I | | HSURT1 chip select input |
| 113 | HSURT1CSO | O | | HSURT1 chip select output |
| 114 | HSURT2SDIO0I | I | HSURT2 | HSURT2 data input 0 |
| 115 | HSURT2SDIO0O | O | | HSURT2 data output 0 |
| 116 | HSURT2SDIO1I | I | | HSURT2 data input 1 |
| 117 | HSURT2SDIO1O | O | | HSURT2 data output 1 |
| 118 | HSURT2SDIO2I | I | | HSURT2 data input 2 |
| 119 | HSURT2SDIO2O | O | | HSURT2 data output 2 |
| 120 | HSURT2SDIO3I | I | | HSURT2 data input 3 |
| 121 | HSURT2SDIO3O | O | | HSURT2 data output 3 |
| 122 | HSURT2SCKI | I | | HSURT2 serial clock input |
| 123 | HSURT2SCKO | O | | HSURT2 serial clock output |
| 124 | HSURT2SDIR | O | | HSURT2 direction indication output |
| 125 | HSURT2CSI | I | | HSURT2 chip select input |
| 126 | HSURT2CSO | O | | HSURT2 chip select output |

Table 2.4 Pin Function Name Definition (4/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 127 | HSURT3SDIO0I | I | HSURT3 | HSURT3 data input 0 |
| 128 | HSURT3SDIO0O | O | | HSURT3 data output 0 |
| 129 | HSURT3SDIO1I | I | | HSURT3 data input 1 |
| 130 | HSURT3SDIO1O | O | | HSURT3 data output 1 |
| 131 | HSURT3SDIO2I | I | | HSURT3 data input 2 |
| 132 | HSURT3SDIO2O | O | | HSURT3 data output 2 |
| 133 | HSURT3SDIO3I | I | | HSURT3 data input 3 |
| 134 | HSURT3SDIO3O | O | | HSURT3 data output 3 |
| 135 | HSURT3SCKI | I | | HSURT3 serial clock input |
| 136 | HSURT3SCKO | O | | HSURT3 serial clock output |
| 137 | HSURT3SDIR | O | | HSURT3 direction indication output |
| 138 | HSURT3CSI | I | | HSURT3 chip select input |
| 139 | HSURT3CSO | O | | HSURT3 chip select output |
| 140 | MTTCAN0RX | I | MTTCAN0 | MTTCAN0 receive data input |
| 141 | MTTCAN0TX | O | | MTTCAN0 transmit data output |
| 142 | MTTCAN0EVT | I | | MTTCAN0 event trigger |
| 143 | MTTCAN0RTP | O | | MTTCAN0 register time mark interrupt pulse |
| 144 | MTTCAN0SOC | O | | MTTCAN0 start of cycle |
| 145 | MTTCAN0SWT | I | | MTTCAN0 stop watch trigger |
| 146 | MTTCAN0TMP | O | | MTTCAN0 trigger time mark interrupt pulse |
| 147 | MCAN0RX | I | MCAN0 | MCAN0 receive data input |
| 148 | MCAN0TX | O | | MCAN0 transmit data output |
| 149 | MCAN1RX | I | MCAN1 | MCAN1 receive data input |
| 150 | MCAN1TX | O | | MCAN1 transmit data output |
| 151 | MCAN2RX | I | MCAN2 | MCAN2 receive data input |
| 152 | MCAN2TX | O | | MCAN2 transmit data output |
| 154 | FLXNTUOUT | O | FLX | NTU output |
| 155 | FLX0RXDA | I | FLX0 | FLX0 channel A receive data input |
| 156 | FLX0RXDB | I | | FLX0 channel B receive data input |
| 157 | FLX0STPWT | I | | FLX0 stop watch trigger input |
| 158 | FLX0TXDA | O | | FLX0 channel A transmit data output |
| 159 | FLX0TXDB | O | | FLX0 channel B transmit data output |
| 160 | FLX0TXENA | O | | FLX0 channel A transmit enable |
| 161 | FLX0TXENB | O | | FLX0 channel B transmit enable |
| 164 | FLX1RXDA | I | FLX1 | FLX1 channel A receive data input |
| 165 | FLX1RXDB | I | | FLX1 channel B receive data input |
| 166 | FLX1STPWT | I | | FLX1 stop watch trigger input |
| 167 | FLX1TXDA | O | | FLX1 channel A transmit data output |
| 168 | FLX1TXDB | O | | FLX1 channel B transmit data output |
| 169 | FLX1TXENA | O | | FLX1 channel A transmit enable |
| 170 | FLX1TXENB | O | | FLX1 channel B transmit enable |
| 173 | SENT0RX | I | SENT0 | SENT ch0 sensor data input |
| 174 | SENT0SPCO | O | | SENT ch0 SPC extension output |

Table 2.4 Pin Function Name Definition (5/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 175 | SENT1RX | I | SENT1 | SENT ch1 sensor data input |
| 176 | SENT1SPCO | O | | SENT ch1 SPC extension output |
| 177 | SENT2RX | I | SENT2 | SENT ch2 sensor data input |
| 178 | SENT2SPCO | O | | SENT ch2 SPC extension output |
| 179 | SENT3RX | I | SENT3 | SENT ch3 sensor data input |
| 180 | SENT3SPCO | O | | SENT ch3 SPC extension output |
| 181 | SENT4RX | I | SENT4 | SENT ch4 sensor data input |
| 182 | SENT4SPCO | O | | SENT ch4 SPC extension output |
| 183 | SENT5RX | I | SENT5 | SENT ch5 sensor data input |
| 184 | SENT5SPCO | O | | SENT ch5 SPC extension output |
| 185 | SENT6RX | I | SENT6 | SENT ch6 sensor data input |
| 186 | SENT6SPCO | O | | SENT ch6 SPC extension output |
| 187 | SENT7RX | I | SENT7 | SENT ch7 sensor data input |
| 188 | SENT7SPCO | O | | SENT ch7 SPC extension output |
| 189 | SENT8RX | I | SENT8 | SENT ch8 sensor data input |
| 190 | SENT8SPCO | O | | SENT ch8 SPC extension output |
| 191 | SENT9RX | I | SENT9 | SENT ch9 sensor data input |
| 192 | SENT9SPCO | O | | SENT ch9 SPC extension output |
| 193 | ETH0COL | I | ETH0 | ETH0 Collision detection signal |
| 194 | ETH0WOL | O | | ETH0 Wake on Lan signal (magic packet detection) |
| 195 | ETH0CRS | I | | ETH0 Carrier detection signal |
| 196 | ETH0MDC | O | | ETH0 Serial management interface transfer clock |
| 197 | ETH0MDI | I | | ETH0 Serial management interface data Input |
| 198 | ETH0MDO | O | | ETH0 Serial management interface data Output |
| 199 | ETH0REF50CK | I | | ETH0 RMII reference clock |
| 200 | ETH0RXCLK | I | | ETH0 Reception clock |
| 201 | ETH0RXD0 | I | | ETH0 Reception data 0 |
| 202 | ETH0RXD1 | I | | ETH0 Reception data 1 |
| 203 | ETH0RXD2 | I | | ETH0 Reception data 2 |
| 204 | ETH0RXD3 | I | | ETH0 Reception data 3 |
| 205 | ETH0RXDV | I | | ETH0 Reception data valid |
| 206 | ETH0RXER | I | | ETH0 Reception data error |
| 207 | ETH0TXCLK | I | | ETH0 Transmission clock |
| 208 | ETH0TXD0 | O | ETH0 | ETH0 Transmission data 0 |
| 209 | ETH0TXD1 | O | | ETH0 Transmission data 1 |
| 210 | ETH0TXD2 | O | | ETH0 Transmission data 2 |
| 211 | ETH0TXD3 | O | | ETH0 Transmission data 3 |
| 212 | ETH0TXEN | O | | ETH0 Transmission data enable |
| 213 | ETH0TXER | O | | ETH0 Transmission data error |
| 214 | ETH0LINKSTA | I | | ETH0 Link status from PHY |

Table 2.4 Pin Function Name Definition (6/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 215 | ETH1COL | I | ETH1 | ETH1 Collision detection signal |
| 216 | ETH1WOL | O | | ETH1 Wake on Lan signal (magic packet detection) |
| 217 | ETH1CRS | I | | ETH1 Carrier detection signal |
| 218 | ETH1MDC | O | | ETH1 Serial management interface transfer clock |
| 219 | ETH1MDI | I | | ETH1 Serial management interface data Input |
| 220 | ETH1MDO | O | | ETH1 Serial management interface data Output |
| 221 | ETH1REF50CK | I | | ETH1 RMII reference clock |
| 222 | ETH1RXCLK | I | | ETH1 Reception clock |
| 223 | ETH1RXD0 | I | | ETH1 Reception data 0 |
| 224 | ETH1RXD1 | I | | ETH1 Reception data 1 |
| 225 | ETH1RXD2 | I | | ETH1 Reception data 2 |
| 226 | ETH1RXD3 | I | | ETH1 Reception data 3 |
| 227 | ETH1RXDV | I | | ETH1 Reception data valid |
| 228 | ETH1RXER | I | | ETH1 Reception data error |
| 229 | ETH1TXCLK | I | | ETH1 Transmission clock |
| 230 | ETH1TXD0 | O | | ETH1 Transmission data 0 |
| 231 | ETH1TXD1 | O | | ETH1 Transmission data 1 |
| 232 | ETH1TXD2 | O | | ETH1 Transmission data 2 |
| 233 | ETH1TXD3 | O | | ETH1 Transmission data 3 |
| 234 | ETH1TXEN | O | | ETH1 Transmission data enable |
| 235 | ETH1TXER | O | | ETH1 Transmission data error |
| 236 | ETH1LINKSTA | I | | ETH1 Link status from PHY |
| 237 | MEMC0A0 | O | External bus | External memory interface address bus bit0 |
| 238 | MEMC0A1 | O | | External memory interface address bus bit1 |
| 239 | MEMC0A2 | O | | External memory interface address bus bit2 |
| 240 | MEMC0A3 | O | | External memory interface address bus bit3 |
| 241 | MEMC0A4 | O | | External memory interface address bus bit4 |
| 242 | MEMC0A5 | O | | External memory interface address bus bit5 |
| 243 | MEMC0A6 | O | | External memory interface address bus bit6 |
| 244 | MEMC0A7 | O | | External memory interface address bus bit7 |
| 245 | MEMC0A8 | O | | External memory interface address bus bit8 |
| 246 | MEMC0D0I | I | | External memory interface data bus bit0 input |
| 247 | MEMC0D1I | I | | External memory interface data bus bit1 input |
| 248 | MEMC0D2I | I | | External memory interface data bus bit2 input |
| 249 | MEMC0D3I | I | | External memory interface data bus bit3 input |
| 250 | MEMC0D4I | I | | External memory interface data bus bit4 input |
| 251 | MEMC0D5I | I | | External memory interface data bus bit5 input |

Table 2.4 Pin Function Name Definition (7/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 252 | MEMC0D6I | I | External bus | External memory interface data bus bit6 input |
| 253 | MEMC0D7I | I | | External memory interface data bus bit7 input |
| 254 | MEMC0D0O | O | | External memory interface data bus bit0 output |
| 255 | MEMC0D1O | O | | External memory interface data bus bit1 output |
| 256 | MEMC0D2O | O | | External memory interface data bus bit2 output |
| 257 | MEMC0D3O | O | | External memory interface data bus bit3 output |
| 258 | MEMC0D4O | O | | External memory interface data bus bit4 output |
| 259 | MEMC0D5O | O | | External memory interface data bus bit5 output |
| 260 | MEMC0D6O | O | | External memory interface data bus bit6 output |
| 261 | MEMC0D7O | O | | External memory interface data bus bit7 output |
| 262 | MEMC0CS0Z | O | | External memory interface chip select signals CS0 |
| 263 | MEMC0CS1Z | O | | External memory interface chip select signals CS1 |
| 264 | MEMC0CS2Z | O | | External memory interface chip select signals CS2 |
| 265 | MEMC0CS3Z | O | | External memory interface chip select signals CS3 |
| 266 | MEMC0RDZ | O | | External memory interface read strobe |
| 267 | MEMC0WRZ | O | | External memory interface write strobe |
| 268 | ADC0CNV | O | ADC0 | ADC0 AD conversion signal |
| 269 | ADC0TRG | I | | ADC0 AD trigger input |
| 270 | ADC0I0 | I | | ADC0 input channel |
| 271 | ADC0I1 | I | | ADC0 input channel |
| 272 | ADC0I2 | I | | ADC0 input channel |
| 273 | ADC0I3 | I | | ADC0 input channel |
| 274 | ADC0I4 | I | | ADC0 input channel |
| 275 | ADC0I5 | I | | ADC0 input channel |
| 276 | ADC0I6 | I | | ADC0 input channel |
| 277 | ADC0I7 | I | | ADC0 input channel |
| 278 | ADC0I8 | I | | ADC0 input channel |
| 279 | ADC0I9 | I | | ADC0 input channel |
| 280 | ADC0I10 | I | | ADC0 input channel |
| 281 | ADC0I11 | I | | ADC0 input channel |
| 282 | ADC0I12 | I | | ADC0 input channel |
| 283 | ADC0I13 | I | | ADC0 input channel |
| 284 | ADC0I14 | I | | ADC0 input channel |
| 285 | ADC0I15 | I | | ADC0 input channel |
| 286 | ADC0I16 | I | | ADC0 input channel |
| 287 | ADC0I17 | I | | ADC0 input channel |
| 288 | ADC0I18 | I | | ADC0 input channel |
| 289 | ADC0I19 | I | | ADC0 input channel |
| 290 | ADC1CNV | O | ADC1 | ADC1 AD conversion signal |
| 291 | ADC1TRG | I | | ADC1 AD trigger input |
| 292 | ADC1I0 | I | | ADC1 input channel |
| 293 | ADC1I1 | I | | ADC1 input channel |
| 294 | ADC1I2 | I | | ADC1 input channel |

Table 2.4 Pin Function Name Definition (8/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 295 | ADC1I3 | I | ADC1 | ADC1 input channel |
| 296 | ADC1I4 | I | | ADC1 input channel |
| 297 | ADC1I5 | I | | ADC1 input channel |
| 298 | ADC1I6 | I | | ADC1 input channel |
| 299 | ADC1I7 | I | | ADC1 input channel |
| 300 | ADC1I8 | I | | ADC1 input channel |
| 301 | ADC1I9 | I | | ADC1 input channel |
| 302 | ADC1I10 | I | | ADC1 input channel |
| 303 | ADC1I11 | I | | ADC1 input channel |
| 304 | ADC1I12 | I | | ADC1 input channel |
| 305 | ADC1I13 | I | | ADC1 input channel |
| 306 | ADC1I14 | I | | ADC1 input channel |
| 307 | ADC1I15 | I | | ADC1 input channel |
| 308 | ADC1I16 | I | | ADC1 input channel |
| 309 | ADC1I17 | I | | ADC1 input channel |
| 310 | ADC1I18 | I | | ADC1 input channel |
| 311 | ADC1I19 | I | | ADC1 input channel |
| 312 | TCK | I | Debug | Debug clock |
| 313 | TDI | I | | Debug data input |
| 314 | TDO | O | | Debug data output |
| 315 | TMS | I | | Debug mode select |
| 316 | RDYZ | O | | Debug ready |
| 317 | TRSTZ | I | | Debug reset |
| 318 | LPDCLK | I | Debug (LPD-4pin) | Debug clock for Low-pin count debug |
| 319 | LPDI | I | | Debug data input for Low-pin count debug |
| 320 | LPDO | O | | Debug data output for Low-pin count debug |
| 321 | LPDCLKOUT | O | | Debug clock output for Low-pin count debug |
| 322 | LPDRSTZ | I | | Debug reset for Low-pin count debug |
| 323 | EVTOZ | O | Nexus | Debug I/F event output signal |
| 324 | EVTIZ | I | | Debug I/F event input signal |
| 325 | FLSCI3TXD | O | Flash Writer I/F | Flash Writer I/F TxD |
| 326 | FLSCI3RXD | I | | Flash Writer I/F RxD |
| 327 | FLSCI3SCKI | I | | Flash Writer I/F SCK |
| 328 | AUDRSTZ | I | AUDR | AUDR reset signal |
| 329 | AUDCK | I | | AUDR clock signal |
| 330 | AUDSYNCZ | I | | AUDR synchronous signal |
| 331 | AUDATA0 | I/O | | AUDR data signal |
| 332 | AUDATA1 | I/O | | AUDR data signal |
| 333 | AUDATA2 | I/O | | AUDR data signal |
| 334 | AUDATA3 | I/O | | AUDR data signal |

Table 2.4 Pin Function Name Definition (9/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 335 | CICREFP | I | Trace I/F | Trace I/F clock pos |
| 336 | CICREFN | I | | Trace I/F clock neg |
| 337 | TODP0 | O | | Trace I/F data pos 0 |
| 338 | TODN0 | O | | Trace I/F data neg 0 |
| 339 | TODP1 | O | | Trace I/F data pos 1 |
| 340 | TODN1 | O | | Trace I/F data neg 1 |
| 341 | MSYNZ | I | | Trace I/F Synchronization request |
| 342 | AURORES1Z | I | | Aurora Domain Reset signal |
| 343 | AURORES2Z | I | | Aurora domain signal Isolation |
| 344 | AURORESPDZ | I | | Core domain Aurora signal Isolation |
| 345 | ERAMRESPDZ | I | | Core domain ERAM signal Isolation |
| 346 | ERAMRES2Z | I | | ERAM domain core signal Isolation |
| 347 | GTM0I0 | I | GTM TIM | GTM TIM 0 input 0 |
| 348 | GTM0I1 | I | | GTM TIM 0 input 1 |
| 349 | GTM0I2 | I | | GTM TIM 0 input 2 |
| 350 | GTM0I3 | I | | GTM TIM 0 input 3 |
| 351 | GTM0I4 | I | | GTM TIM 0 input 4 |
| 352 | GTM0I5 | I | | GTM TIM 0 input 5 |
| 353 | GTM0I6 | I | | GTM TIM 0 input 6 |
| 354 | GTM0I7 | I | | GTM TIM 0 input 7 |
| 355 | GTM1I0 | I | | GTM TIM 1 input 0 |
| 356 | GTM1I1 | I | | GTM TIM 1 input 1 |
| 357 | GTM1I2 | I | | GTM TIM 1 input 2 |
| 358 | GTM1I3 | I | | GTM TIM 1 input 3 |
| 359 | GTM1I4 | I | | GTM TIM 1 input 4 |
| 360 | GTM1I5 | I | | GTM TIM 1 input 5 |
| 361 | GTM1I6 | I | | GTM TIM 1 input 6 |
| 362 | GTM1I7 | I | | GTM TIM 1 input 7 |
| 363 | GTMAT0O0 | O | ATOM | GTM ATOM 0 output 0 |
| 364 | GTMAT0O1 | O | | GTM ATOM 0 output 1 |
| 365 | GTMAT0O2 | O | | GTM ATOM 0 output 2 |
| 366 | GTMAT0O3 | O | | GTM ATOM 0 output 3 |
| 367 | GTMAT0O4 | O | | GTM ATOM 0 output 4 |
| 368 | GTMAT0O5 | O | | GTM ATOM 0 output 5 |
| 369 | GTMAT0O6 | O | | GTM ATOM 0 output 6 |
| 370 | GTMAT0O7 | O | | GTM ATOM 0 output 7 |
| 371 | GTMAT1O0 | O | | GTM ATOM 1 output 0 |
| 372 | GTMAT1O1 | O | | GTM ATOM 1 output 1 |
| 373 | GTMAT1O2 | O | | GTM ATOM 1 output 2 |
| 374 | GTMAT1O3 | O | | GTM ATOM 1 output 3 |
| 375 | GTMAT1O4 | O | | GTM ATOM 1 output 4 |
| 376 | GTMAT1O5 | O | | GTM ATOM 1 output 5 |
| 377 | GTMAT1O6 | O | | GTM ATOM 1 output 6 |
| 378 | GTMAT1O7 | O | | GTM ATOM 1 output 7 |

Table 2.4 Pin Function Name Definition (10/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 379 | GTMAT2O0 | O | ATOM | GTM ATOM 2 output 0 |
| 380 | GTMAT2O1 | O | | GTM ATOM 2 output 1 |
| 381 | GTMAT2O2 | O | | GTM ATOM 2 output 2 |
| 382 | GTMAT2O3 | O | | GTM ATOM 2 output 3 |
| 383 | GTMAT2O4 | O | | GTM ATOM 2 output 4 |
| 384 | GTMAT0O0N | O | DTM | GTM ATOM 0 output 0 neg |
| 385 | GTMAT0O1N | O | | GTM ATOM 0 output 1 neg |
| 386 | GTMAT0O2N | O | | GTM ATOM 0 output 2 neg |
| 387 | GTMAT0O3N | O | | GTM ATOM 0 output 3 neg |
| 388 | GTMAT1O0N | O | | GTM ATOM 1 output 0 neg |
| 389 | GTMAT1O1N | O | | GTM ATOM 1 output 1 neg |
| 390 | GTMAT1O2N | O | | GTM ATOM 1 output 2 neg |
| 391 | GTMAT1O3N | O | | GTM ATOM 1 output 3 neg |
| 392 | GTMAT2O0N | O | | GTM ATOM 2 output 0 neg |
| 393 | GTMAT2O1N | O | | GTM ATOM 2 output 1 neg |
| 394 | GTMAT2O2N | O | | GTM ATOM 2 output 2 neg |
| 395 | GTMAT2O3N | O | | GTM ATOM 2 output 3 neg |
| 396 | ESO0Z | I | GTM Hi-Z control | Emergency shut-off 0 |
| 397 | ESO1Z | I | | Emergency shut-off 1 |
| 398 | ESO2Z | I | | Emergency shut-off 2 |
| 399 | ICUMGPIO0 | O | ICUMC | ICUMC GPIO control 0 |
| 400 | ICUMGPIO1 | O | | ICUMC GPIO control 1 |
| 401 | BHPDGRCLK0 | O | BHP | Degrading clock output for MCAN |
| 402 | BHPDGRCLK1 | O | | Degrading clock output for FlexRay |
| 403 | BHPDGREN | I | | Degrading allowed pin |
| 404 | ERROROUTZ | O | ECM | Error output signal |
| 405 | CVMOUTZ | O | System Control | CVM internal voltage error detection output signal |
| 406 | FLMD0 | I | | Operating mode select pin |
| 407 | FLMD1 | I | | Operating mode select pin |
| 408 | MODE0 | I | | Operating mode select pin |
| 409 | MODE1 | I | | Operating mode select pin |
| 410 | RESETZ | I | | Reset input |
| 411 | RESETOUTZ | O | | Reset output |
| 412 | X1 | I | | Main oscillator resonator connections |
| 413 | X2 | O | | Main oscillator resonator connections |
| 414 | JP0_0 | I/O | JP0 | JTAG Port group 0 port0 |
| 415 | JP0_1 | I/O | | JTAG Port group 0 port1 |
| 416 | JP0_2 | I/O | | JTAG Port group 0 port2 |
| 417 | JP0_3 | I/O | | JTAG Port group 0 port3 |
| 418 | JP0_4 | I | | JTAG Port group 0 port4 |
| 419 | JP0_5 | I/O | | JTAG Port group 0 port5 |

Table 2.4 Pin Function Name Definition (11/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 420 | P0_0 | I/O | P0 | Port group 0 port0 |
| 421 | P0_1 | I/O | | Port group 0 port1 |
| 422 | P0_2 | I/O | | Port group 0 port2 |
| 423 | P0_3 | I/O | | Port group 0 port3 |
| 424 | P0_4 | I/O | | Port group 0 port4 |
| 425 | P0_5 | I/O | | Port group 0 port5 |
| 426 | P0_6 | I/O | | Port group 0 port6 |
| 427 | P0_7 | I/O | | Port group 0 port7 |
| 428 | P0_8 | I/O | | Port group 0 port8 |
| 429 | P0_9 | I/O | | Port group 0 port9 |
| 430 | P0_10 | I/O | | Port group 0 port10 |
| 431 | P0_13 | I/O | | Port group 0 port13 |
| 432 | P0_14 | I/O | | Port group 0 port14 |
| 433 | P1_1 | I/O | P1 | Port group 1 port1 |
| 434 | P1_2 | I/O | | Port group 1 port2 |
| 435 | P1_3 | I/O | | Port group 1 port3 |
| 436 | P1_4 | I/O | | Port group 1 port4 |
| 437 | P1_5 | I/O | | Port group 1 port5 |
| 438 | P1_6 | I/O | | Port group 1 port6 |
| 439 | P1_7 | I/O | | Port group 1 port7 |
| 440 | P2_0 | I/O | P2 | Port group 2 port0 |
| 441 | P2_1 | I/O | | Port group 2 port1 |
| 442 | P2_2 | I/O | | Port group 2 port2 |
| 443 | P2_3 | I/O | | Port group 2 port3 |
| 444 | P2_4 | I/O | | Port group 2 port4 |
| 445 | P2_5 | I/O | | Port group 2 port5 |
| 446 | P2_6 | I/O | | Port group 2 port6 |
| 447 | P2_7 | I/O | | Port group 2 port7 |
| 448 | P2_8 | I/O | | Port group 2 port8 |
| 449 | P2_9 | I/O | | Port group 2 port9 |
| 450 | P2_10 | I/O | | Port group 2 port10 |
| 451 | P2_11 | I/O | | Port group 2 port11 |
| 452 | P2_12 | I/O | | Port group 2 port12 |
| 453 | P2_13 | I/O | | Port group 2 port13 |
| 454 | P2_14 | I/O | | Port group 2 port14 |
| 455 | P2_15 | I/O | | Port group 2 port15 |
| 456 | P3_0 | I/O | P3 | Port group 3 port0 |
| 457 | P3_1 | I/O | | Port group 3 port1 |
| 458 | P3_2 | I/O | | Port group 3 port2 |
| 459 | P3_3 | I/O | | Port group 3 port3 |
| 460 | P3_4 | I/O | | Port group 3 port4 |
| 461 | P3_5 | I/O | | Port group 3 port5 |
| 462 | P3_6 | I/O | | Port group 3 port6 |
| 463 | P3_7 | I/O | | Port group 3 port7 |

Table 2.4 Pin Function Name Definition (12/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 464 | P3_8 | I/O | P3 | Port group 3 port8 |
| 465 | P3_9 | I/O | | Port group 3 port9 |
| 466 | P3_10 | I/O | | Port group 3 port10 |
| 467 | P3_11 | I/O | | Port group 3 port11 |
| 468 | P3_12 | I/O | | Port group 3 port12 |
| 469 | P3_13 | I/O | | Port group 3 port13 |
| 470 | P3_14 | I/O | | Port group 3 port14 |
| 471 | P4_0 | I/O | P4 | Port group 4 port0 |
| 472 | P4_1 | I/O | | Port group 4 port1 |
| 473 | P4_2 | I/O | | Port group 4 port2 |
| 474 | P4_3 | I/O | | Port group 4 port3 |
| 475 | P4_4 | I/O | | Port group 4 port4 |
| 476 | P4_5 | I/O | | Port group 4 port5 |
| 477 | P4_6 | I/O | | Port group 4 port6 |
| 478 | P4_7 | I/O | | Port group 4 port7 |
| 479 | P4_8 | I/O | | Port group 4 port8 |
| 480 | P4_9 | I/O | | Port group 4 port9 |
| 481 | P4_10 | I/O | | Port group 4 port10 |
| 482 | P4_11 | I/O | | Port group 4 port11 |
| 483 | P4_12 | I/O | | Port group 4 port12 |
| 484 | P4_13 | I/O | | Port group 4 port13 |
| 485 | P4_14 | I/O | | Port group 4 port14 |
| 486 | P5_0 | I/O | P5 | Port group 5 port0 |
| 487 | P5_1 | I/O | | Port group 5 port1 |
| 488 | P5_4 | I/O | | Port group 5 port4 |
| 489 | P5_5 | I/O | | Port group 5 port5 |
| 490 | P5_6 | I/O | | Port group 5 port6 |
| 491 | P5_7 | I/O | | Port group 5 port7 |
| 492 | P5_8 | I/O | | Port group 5 port8 |
| 493 | P5_9 | I/O | | Port group 5 port9 |
| 494 | P5_10 | I/O | | Port group 5 port10 |
| 495 | P5_11 | I/O | | Port group 5 port11 |
| 496 | P5_12 | I/O | | Port group 5 port12 |
| 497 | P5_13 | I/O | | Port group 5 port13 |
| 498 | P5_14 | I/O | | Port group 5 port14 |
| 499 | P5_15 | I/O | | Port group 5 port15 |
| 500 | P6_0 | I/O | P6 | Port group 6 port0 |
| 501 | P6_1 | I/O | | Port group 6 port1 |
| 502 | P6_2 | I/O | | Port group 6 port2 |
| 503 | P6_3 | I/O | | Port group 6 port3 |
| 504 | P6_4 | I/O | | Port group 6 port4 |
| 505 | P6_5 | I/O | | Port group 6 port5 |
| 506 | P6_6 | I/O | | Port group 6 port6 |
| 507 | P6_7 | I/O | | Port group 6 port7 |

Table 2.4 Pin Function Name Definition (13/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 508 | P6_8 | I/O | P6 | Port group 6 port8 |
| 509 | P6_9 | I/O | | Port group 6 port9 |
| 510 | P6_10 | I/O | | Port group 6 port10 |
| 511 | P6_11 | I/O | | Port group 6 port11 |
| 512 | P6_12 | I/O | | Port group 6 port12 |
| 513 | P6_13 | I/O | | Port group 6 port13 |
| 514 | P6_14 | I/O | | Port group 6 port14 |
| 515 | P6_15 | I/O | | Port group 6 port15 |
| 516 | P7_0 | I/O | P7 | Port group 7 port0 |
| 517 | P7_1 | I/O | | Port group 7 port1 |
| 518 | P7_2 | I/O | | Port group 7 port2 |
| 519 | P7_3 | I/O | | Port group 7 port3 |
| 520 | P7_4 | I/O | | Port group 7 port4 |
| 521 | P7_5 | I/O | | Port group 7 port5 |
| 522 | P7_6 | I/O | | Port group 7 port6 |
| 523 | P7_7 | I/O | | Port group 7 port7 |
| 524 | P7_8 | I/O | | Port group 7 port8 |
| 525 | P7_9 | I/O | | Port group 7 port9 |
| 526 | P8_0 | I/O | P8 | Port group 8 port0 |
| 527 | P8_1 | I/O | | Port group 8 port1 |
| 528 | P8_2 | I/O | | Port group 8 port2 |
| 529 | P8_3 | I/O | | Port group 8 port3 |
| 530 | P8_4 | I/O | | Port group 8 port4 |
| 531 | P8_5 | I/O | | Port group 8 port5 |
| 532 | P8_6 | I/O | | Port group 8 port6 |
| 533 | P8_7 | I/O | | Port group 8 port7 |
| 534 | P8_8 | I/O | | Port group 8 port8 |
| 535 | P8_9 | I/O | | Port group 8 port9 |
| 536 | P8_10 | I/O | | Port group 8 port10 |
| 537 | P8_11 | I/O | | Port group 8 port11 |
| 538 | P8_12 | I/O | | Port group 8 port12 |
| 539 | P8_13 | I/O | | Port group 8 port13 |
| 540 | P8_14 | I/O | | Port group 8 port14 |
| 541 | P8_15 | I/O | | Port group 8 port15 |
| 542 | P9_0 | I/O | P9 | Port group 9 port0 |
| 543 | P9_1 | I/O | | Port group 9 port1 |
| 544 | P9_2 | I/O | | Port group 9 port2 |
| 545 | P9_3 | I/O | | Port group 9 port3 |
| 546 | P9_4 | I/O | | Port group 9 port4 |
| 547 | P9_5 | I/O | | Port group 9 port5 |
| 548 | P9_6 | I/O | | Port group 9 port6 |
| 549 | P9_7 | I/O | | Port group 9 port7 |
| 550 | P9_8 | I/O | | Port group 9 port8 |

Table 2.4 Pin Function Name Definition (14/14)

| Line | Function name | I/O | Category | Explanation |
|---|---|---|---|---|
| 551 | EnVCC | power | IO_Power | IO supply voltage n = 0, 1 |
| 552 | EnVSS | power | | IO ground n = 0, 1 |
| 553 | VCC | power | Internal Regulator | Regulator supply voltage |
| 554 | VDD | power | Core_Power | Core supply voltage |
| 555 | VSS | power | Core_Ground | Core ground |
| 556 | DVCC | power | Trace_IF_Power | Aurora I/F supply voltage (Analog) |
| 557 | DVDD | power | | Aurora I/F supply voltage (Digital) |
| 558 | DVSS | power | | Aurora I/F ground |
| 559 | AnVCC | power | ADC | ADC supply voltage n = 0, 1 |
| 560 | AnVSS | power | | ADC ground n = 0, 1 |
| 561 | AnVREFH | power | | ADC reference voltage plus n = 0, 1 |
| 562 | EMUVCC | power | EMU | Aurora control IO supply voltage |
| 563 | EMUVDD | power | | Aurora control core supply voltage |
| 564 | EMUVSS | power | | Aurora control ground |
| 565 | SYSVCC | power | System | System control supply voltage |
| 566 | OSCVSS | power | OSC | OSC ground |
| 567 | ERAMVDD | power | ERAM Control | ERAM core supply voltage |
| 568 | ERAMnVCC | power | | ERAM IO supply voltage n = 0, 1 |
| 569 | ERAMnVSS | power | | ERAM ground n = 0, 1 |

## 2.2.5 Recommended Connection of Unused Pins

**Table 2.5** Recommended Connection of Unused Pins

**Table 2.5 Recommended Connection of Unused Pins (1/3)**

| Category | Pin Name | Recommended Connection of Unused Pins |
|---|---|---|
| CLOCK | X1 | (Must be used.) |
| | X2 | Leave open-circuit in case of external clock supply. |
| SYSTEM | RESETZ | (Must be used.) |
| | FLMD0 | (Must be used.) |
| | P4_2, P4_3 | • When FLMD0 is H level and P4_5(FLMD1) is H level<br>Connected to E1VSS via a resistor.<br>• When FLMD0 is L level or P4_5(FLMD1) is L level<br>[Buffer disabled]<br>Leave open-circuit.<br>[Input]<br>Leave open-circuit and the settings are to enable on-chip pullup/pull-down resistors.(use PUn_m, PDn_m)<br>Connected to E1VCC or E1VSS via a resistor.<br>[Output]<br>Leave open-circuit. |
| SYSTEM | P4_5 | • When FLMD0 is H level<br>Connected to E1VSS via a resistor.<br>• When FLMD0 is L level<br>[Buffer disabled]<br>Leave open-circuit.<br>[Input]<br>Leave open-circuit and the settings are to enable on-chip pullup/pull-down resistors.(use PUn_m, PDn_m)<br>Connected to E1VCC or E1VSS via a resistor.<br>[Output]<br>Leave open-circuit. |
| Pn_m | P0<br>P1<br>P2<br>P3<br>P4(excluding P4_2, P4_3, and P4_5)<br>P5<br>P6<br>P7<br>P8<br>P9 | [Buffer disabled]<br>Leave open-circuit.<br>[Input]<br>Leave open-circuit and the settings are to enable on-chip pullup/pull-down resistors.(use PUn_m, PDn_m)<br>Connected to EnVCC or EVSS via a resistor.<br>[Output]<br>Leave open-circuit. |

**Table 2.5 Recommended Connection of Unused Pins (2/3)**

| Category | Pin Name | Recommended Connection of Unused Pins |
|---|---|---|
| JP0_m | JP0_0 : TDI/LPDI | Connected to E0VCC via a resistor*1 |
| | JP0_0 : JP0_0 | [Buffer disabled]<br>Leave open-circuit.<br>[Input]<br>Leave open-circuit and the settings are to enable on-chip pullup/pull-down resistors.(use PUn_m, PDn_m)<br>Connected to E0VCC or E0VSS via a resistor.*3<br>[Output]<br>Leave open-circuit. |
| | JP0_1 : TDO/ LPDO | Leave open-circuit. |
| | JP0_1 : JP0_1 | [Buffer disabled]<br>Leave open-circuit.<br>[Input]<br>Leave open-circuit and the settings are to enable on-chip pullup/pull-down resistors.(use PUn_m, PDn_m)<br>Connected to E0VCC or E0VSS via a resistor.*3<br>[Output]<br>Leave open-circuit. |
| | JP0_2 : TCK/ LPDCLK | Connected to E0VCC via a resistor*1, or Leave open-circuit. |
| | JP0_2 : JP0_2 | [Buffer disabled]<br>Leave open-circuit.<br>[Input]<br>Leave open-circuit and the settings are to enable on-chip pullup/pull-down resistors.(use PUn_m, PDn_m)<br>Connected to E0VCC or E0VSS via a resistor.*3<br>[Output]<br>Leave open-circuit. |
| | JP0_3 : TMS | Connected to E0VCC via a resistor*1 |
| | JP0_3 : JP0_3 | [Buffer disabled]<br>Leave open-circuit.<br>[Input]<br>Leave open-circuit and the settings are to enable on-chip pullup/pull-down resistors.(use PUn_m, PDn_m)<br>Connected to E0VCC or E0VSS via a resistor.*3<br>[Output]<br>Leave open-circuit. |
| | JP0_4 : TRSTZ/ LPDRSTZ/JP0_4 | Connected to E0VSS via a resistor*2 |
| | JP0_5 : RDYZ/ LPDCLKOUT | Leave open-circuit. |
| | JP0_5 : JP0_5 | [Buffer disabled]<br>Leave open-circuit.<br>[Input]<br>Leave open-circuit and the settings are to enable on-chip pullup/pull-down resistors.(use PUn_m, PDn_m)<br>Connected to E0VCC or E0VSS via a resistor.<br>[Output]<br>Leave open-circuit. |
| ADC | ADC0In | Leave open-circuit<br>Or connected to A0VCC or A0VSS . |
| | ADC1In | Leave open-circuit<br>Or connected to A1VCC or A1VSS . |
| Other Func | ERROROUTZ | Leave open-circuit |
| | CVMOUTZ | Leave open-circuit |

**Table 2.5 Recommended Connection of Unused Pins (3/3)**

| Category | Pin Name | Recommended Connection of Unused Pins |
|---|---|---|
| Trace I/F | AURORES1Z | Connected to EMUVSS via a resistor. |
| | AURORES2Z | |
| | AURORESPDZ | Connected to E0VSS via a resistor. |
| | CICREFN, CLCREFP | Leave open-circuit |
| | TODN0, TODN1 | Leave open-circuit |
| | TODP0, TODP1 | |
| | MSYNZ | Connected to E0VCC via a resistor. |
| ERAM | ERAMRESPDZ | Connected to E1VSS via a resistor. |
| | ERAMRES2Z | Connected to ERAM1VSS via a resistor. |
| EMU | EVTOZ | Leave open-circuit |
| | EVTIZ | Connected to E1VCC via a resistor. |
| | AUDATA0 to AUDATA3 | Connected to E1VCC via a resistor. |
| | AUDSYNCZ | Connected to E1VCC via a resistor. |
| | AUDCK | Connected to E1VCC via a resistor. |
| | AUDRSTZ | Connected to E1VSS via a resistor. |
| Power | EnVCC, EnVSS | (Must be used.) |
| | OSCVSS | (Must be used.) |
| | SYSVCC | (Must be used.) |
| | VDD, VSS | (Must be used.) |
| | VCC | (Must be used.) |
| | A0VREFH, A1VREFH | Connected to EnVCC. |
| | A0VCC, A1VCC | Connected to EnVCC. |
| | A0VSS, A1VSS | Connected to EnVSS. |
| | DVCC, DVDD, DVSS | (Must be used.) for Emulation devices |
| | EMUVCC, EMUVDD, EMUVSS | (Must be used.) for Emulation devices |
| | ERAMnVCC, ERAMVDD, ERAMnVSS | (Must be used.) for Emulation devices |

Note 1. Except for P1H-CE. P1H-CE is “Connected to E1VCC via a resistor.”

Note 2. Except for P1H-CE. P1H-CE is “Connected to E1VSS via a resistor.”

Note 3. Except for P1H-CE. P1H-CE is “Connected to E1VCC or E1VSS via a resistor.”

## 2.3 Port Overview

This document defines the standard “port control logic” including the interface between IO buffers. Each parameters as Port Number are optimized for RH850/P1x-C products.

### 2.3.1 Introduction

The port of microcontroller is an interface between external devices and the internal resources (IPs) of the microcomputer. From design point of view, a port usually consists of IO buffer and the control logic as it is shown below.

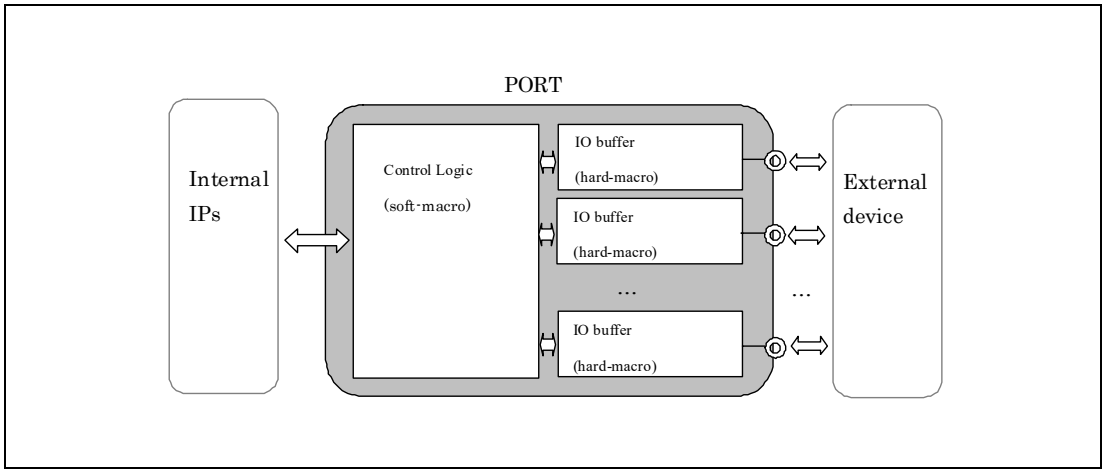



**Figure 2.7 Port structure overview**

### 2.3.2 Functional Overview

Active microcontroller reset result in a Hi-Z state for nearly all port pins. For exceptions refer to **Table 2.2, Pin Status**. After RESETZ release SW can define different state. All state changes by SW become active immediately after bus cycle of the associated write access is finished (<1us).

When a port pin is configured as digital input it is possible to configure it with pull-up or pull-down or without pull up. Internal pull-up on an input pin is enabled by setting pull-up register (PUn.PUn_m = 1) and pull-down is enabled by setting pull-down register (PDn.PDn_m = 1). If a pin is configured such that both an internal pull-up resistor (PUn.PUn_m = 1) and pull-down resistor (PDn.PDn_m = 1) are connected, the pull-down resistor is automatically selected and the pull-up resistor is not connected. If no pull-up or pull-down is required on an input port pin deselect pull-up and pull-down register (PUn.PUn_m = 0, PDn.PDn_m = 0).

#### External Interrupts Pins:

The maskable external interrupt pins (INTP0 to INTPn) and one non maskable interrupt pin (NMI) are available in the device. For external interrupts, falling edge, rising edge or both edges is selectable for each pin by the setting of the External Interrupt Detecting Method Select bits in the external interrupt control register. Furthermore, interrupt priority levels 0 to 15 can be set for each pin in interrupt control registers (IC0 to ICn). When edge detection is selected as an interrupt detecting method, an interrupt request is detected by a change in INTPn pins, and an interrupt request signal is sent to the INTC. Four external interrupts can generate DMA request and for more details see **Section 6, Interrupt Controller (INTC)**.

#### 2.3.2.1 Port Category

(1) Numeric Port

The numeric port provides functions including Digital IO and alternative functions.

For example, functions like timer outputs, serials communications can be assigned on this port. External bus interface can also be assigned.

(2) JTAG Port

Several special functionalities are provided by JTAG port. This port is standardized to be used as a test interface, a debug interface, and port interface including flash programming interface.

The switch between debug interface and port mode is controlled by OPBT2. (See **Section 32.12.1, OPBT0 — Option Byte 0**)

Although any port functionalities can be realized with this pin, they cannot be emulated when the pin is used as a debug interface. Therefore it is recommended not to assign important user functions to this pin.

### 2.3.3 Operation Mode

Pins can operate in three operation modes.

- Port mode(PMCn.PMCn_m bit = 0)

  A pin in port mode operates as a general purpose input/output pin. The input / output mode is selected by setting the PMn.PMn_m bit.

- Alternative mode software I/O control(PMCn.PMCn_m bit = 1, PIPCn.PIPCn_m bit = 0)

  In this mode, the pins operate as alternative functions. The input / output mode is selected by setting the PMn.PMn_m bit.

- Alternative mode direct I/O control(PMCn.PMCn_m bit = 1, PIPCn.PIPCn_m bit = 1)

  In this mode, the pins operate as alternative funtions. Unlike the alternative mode software I/O control, however, the input / output direction is selected by the alternative function.

The following is register effect related to operation modes and pin I/O direction.

- PMCn.PMCn_m bit

  This bit selects port mode(PMCn_m = 0) or alternative mode(PMCn_m = 1).

- PMn.PMn_m bit

  This bit selects input(PMn_m = 1) or output(PMn_m = 0) when the port mode(PMCn_m = 0) and alternative mode software I/O control(PMCn_m = 1, PIPCn_m = 0) have been selected.

- PIBCn.PIBCn_m bit

  This bit disables(PIBCn_m = 0) or enables(PIBCn_m = 1) the input buffer in input port mode(PMCn_m = 0, PMn_m = 1). If alternative mode, this bit must always set to 0.

- PIPCn.PIPCn_m bit

  This bit selects alternative mode software I/O control or alternative mode direct I/O control.

- PBDCn.PBDCn_m bit

  In output mode or output enabled by alternative function, when this bit is set to 1, the pin enters the bidirectional mode.

The input / output direction and each modes by register setting is shown below Table 2.x Pin register setting

**Table 2.6 Pin register setting**

| PMCn_m | PMn_m | PIBCn_m | PIPCn_m | PBDCn_m | Modes | I/O Direction |
|---|---|---|---|---|---|---|
| 0 | 0 | X | X | 0 | Port mode<br>Output mode | Output |
| | | | | 1 | Port mode<br>Bi-directional mode | Input / Output |
| | 1 | 0 | | X | Port mode<br>Input mode (input disabled) | - |
| | | 1 | | | Port mode<br>Input mode (input enabled) | Input |
| 1 | 0 | 0*1 | 0 | 0 | Alternative mode<br>Software I/O control<br>Output mode | Output |
| | | | | 1 | Alternative mode<br>Software I/O control<br>Bi-directional mode | Input / Output |
| | 1 | | | X | Alternative mode<br>Software I/O control<br>Input mode | Input |
| | X | | 1 | 0 | Alternative mode<br>Direct I/O control<br>Input or output | Controlled by the alternative function |
| | | | | 1 | Alternative mode<br>Direct I/O control<br>Input or bi-direction | Controlled by the alternative function |

Note 1. Setting PIBC 1 in alternative mode is prohibited.

**Table 2.7 Alternative mode selection table**

| PFCEn_[15:0] | PFCn_[15:0] | PMn_[15:0] | Function |
|---|---|---|---|
| 0 | 0 | 1 | Alternative peripheral function 1 (Alternative Mode 1) Input |
| | | 0 | Alternative peripheral function 1 (Alternative Mode 1) Output |
| | 1 | 1 | Alternative peripheral function 2 (Alternative Mode 2) Input |
| | | 0 | Alternative peripheral function 2 (Alternative Mode 2) Output |
| 1 | 0 | 1 | Alternative peripheral function 3 (Alternative Mode 3) Input |
| | | 0 | Alternative peripheral function 3 (Alternative Mode 3) Output |
| | 1 | 1 | Alternative peripheral function 4 (Alternative Mode 4) Input |
| | | 0 | Alternative peripheral function 4 (Alternative Mode 4) Output |

Remark:
Some input or output functions is assigned to more than one pin. For using functions on multiple pins refer to the related register description in **Section 2.5.6, PFCEn/JPFCE0 — Port Function Control Expansion Register**.

**CAUTION**

**To prevent malfunction, secure reset value to registers that are not available for the individual product. For port availability of each product see Table 2.1.**

### 2.3.4 Pin data input/output

The registers used for data input/output are described below.

The location that is read via the PPRn register differs depending on the pin mode.

(1) Output data

In the port mode (PMCn.PMCn_m = 0), the value of the Pn.Pn_m bit is output to the Pn_m pin.

(2) Input data

When the PPRn register is read, either the value of the Pn_m pin, the value of the corresponding bit of the port register Pn.Pn_m, or the value output by the alternative function is returned.

Which value is returned depends on the pin mode and setting of several control bits. The different PPRn read modes are shown in the table 2.x PPRn_m Read Values.

**Table 2.8 PPRn_m Read Values**

| PMC n_m | PMn_m | PIBC n_m | PIPC n_m | PBDC n_m | Mode | PPRn_m Read Value |
|---|---|---|---|---|---|---|
| 0 | 0 | X | X | 0 | Port Mode<br>Output Mode | Pn.Pn_m bit |
| | | | | 1 | Port Mode<br>Bi-directional Mode | Pn_m pin |
| | 1 | 0 | | X | Port Mode<br>Input Mode,<br>(Input disabled) | Pn.Pn_m bit |
| | | 1 | | | Port Mode<br>Input Mode<br>(Input enabled) | Pn_m pin |
| 1 | 0 | 0*1 | 0 | 0 | Alternative Mode<br>Software I/O control<br>Output Mode | Alternative-function internal output signal |
| | | | | 1 | Alternative Mode<br>Software I/O control<br>Bi-directional Mode | Pn_m pin |
| | 1 | | | X | Alternative Mode<br>Software I/O control<br>Input Mode | Pn_m pin |
| | X | | 1 | 0 | Alternative Mode<br>Direct I/O control<br>Input or output | I/O port in alternative mode:<br>* Input: Pn_m pin<br>* Output: Alternative-function internal output signal |
| | | | | 1 | Alternative Mode<br>Direct I/O control<br>Input or bi-direction | I/O port in alternative mode:<br>* Input: Pn_m pin<br>* Bi-direction: Pn_m pin |

Note 1. Setting PIBC 1 in alternative mode is prohibited.

## 2.4 Port Functions

This section explains port control logic functions.

### 2.4.1 Port Control Logic Block Diagram

This section describes the superset of port structure and functionality. A port in this device is a subset of the logic structure shown below.

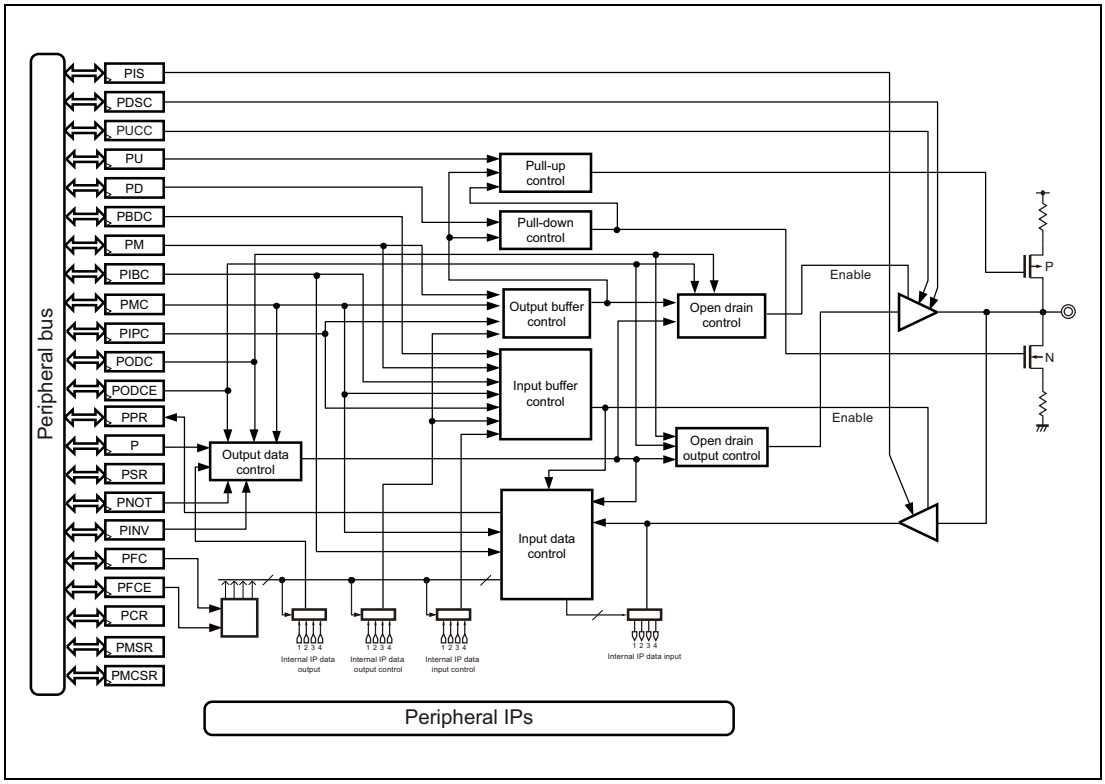



**Figure 2.8 Port Control Logic Block Diagram**

## 2.4.2 JTAG port 0 (JP0)

### 2.4.2.1 Control Register

| Port | Register | Function | Access Size | Base Address | Offset Address | Reset Value | Existent Bit | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | Position |
| JP0 | JP0 | JTAG Port register 0 | 8 | FFC2 $0000_H$ | $0000_H$ | $00_H$ | R/W | 5, 3-0 |
| | JPSR0 | JTAG Set Reset register 0 | 32 | | $0004_H$ | 0000 $0000_H$ | R/W | 21, 19-16, 5, 3-0 |
| | JPNOT0 | JTAG Port NOT register 0 | 8 | | $0008_H$ | $00_H$ | W | 5, 3-0 |
| | JPPR0 | JTAG Port Pin Read register 0 | 8 | | $000C_H$ | $00_H$ | R | 5-0 |
| | JPM0 | JTAG Port Mode register 0 | 8 | | $0010_H$ | $FF_H$ | R/W | 5, 3-0 |
| | JPMC0 | JTAG Port Mode Control register 0 | 8 | | $0014_H$ | $00_H$ | R/W | 2-0 |
| | JPFCE0 | JTAG Port Function Control Expansion register 0 | 8 | | $001C_H$ | $00_H$ | R/W | 2-0 |
| | JPMSR0 | JTAG Port Mode Set Reset register 0 | 32 | | $0020_H$ | 0000 $00FF_H$ | R/W | 21, 19-16, 5, 3-0 |
| | JPMCSR0 | JTAG Port Mode Control Set Reset register 0 | 32 | | $0024_H$ | 0000 $0000_H$ | R/W | 18-16, 2-0 |
| | JPINV0 | JTAG Port output value Inversion register 0 | 8 | | $0030_H$ | $00_H$ | R/W | 5, 3-0 |
| | JPIBC0 | JTAG Port Port Input Buffer Control register 0 | 8 | | $4000_H$ | $00_H$ | R/W | 5-0 |
| | JPBDC0 | JTAG Port Bi-Direction Control register 0 | 8 | | $4004_H$ | $00_H$ | R/W | 5, 3-0 |
| | JPU0 | JTAG Pull-up option register 0 | 8 | | $400C_H$ | $00_H$ | R/W | 5, 3-0 |
| | JPD0 | JTAG Pull-down option register 0 | 8 | | $4010_H$ | $00_H$ | R/W | 5, 3-0 |
| | JPODC0 | JTAG Port Open Drain Control register 0 | 32 | | $4014_H$ | 0000 $0000_H$ | R/W | 5, 3-0 |
| | JPDSC0 | JTAG Port Drive Strength Control register 0 | 32 | | $4018_H$ | 0000 $0000_H$ | R/W | 5, 3-0 |
| | JPIS0 | JTAG Port Input Selection register 0 | 8 | | $401C_H$ | $00_H$ | R/W | 5, 3-0 |
| | JPUCC0 | JTAG Port Universal Characteristics Control register 0 | 32 | | $4028_H$ | 0000 $0000_H$ | R/W | 5, 3-0 |
| | JPODCE0 | JTAG Port Open Drain Control Expansion register 0 | 32 | | $4038_H$ | 0000 $0000_H$ | R/W | 5, 3-0 |

**Note:** Not all bit exist for all products. For details to the available port pins refer to **Table 2.1**

Note 1. When the JP0_m pin are used as NEXUS or LPD-4pin, set Reset Value to the register of JPM0, JPMC0 and JPIBC0.

The change between debug interface and Port mode of the JTAG port is controlled by the OPBT2. (See **Section 32.12.3, OPBT2 — Option Byte 2**) The following table describes the functionality of each JP0_m pin in the different modes.

| OPJTAG1 | OPJTAG0 | Mode | JP0_0 | JP0_1 | JP0_2 | JP0_3 | JP0_4 | JP0_5 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Nexus (JTAG) | TDI input | TDO output | TCK input | TMS input | TRSTZ input | RDYZ output |
| 0 | 1 | LPD (4-pin) | LPDI input | LPDO output | LPDCLK input | Port | LPDRSTZ input | LPDCLKOUT output |
| 0 | 0 | GPIO | Port | Port | Port | Port | TRSTZ Input*1 | Port |

Note 1. JP0_4 pin level can only be read by JPPR0 register in Port mode.

## 2.4.3 Port 0 (P0)

### 2.4.3.1 Control Register

| Port | Register | Function | Access Size | Base Address | Offset Address | Reset Value | Existent Bit | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | Position |
| P0 | P0 | Port register 0 | 16 | FFC1 $0000_H$ | $0000_H$ | $0000_H$ | R/W | 14, 13, 10-0 |
| | PSR0 | Port Set Reset register 0 | 32 | | $0004_H$ | $0000\ 0000_H$ | R/W | 30, 29, 26-16, 14, 13, 10-0 |
| | PNOT0 | Port NOT register 0 | 16 | | $0008_H$ | $0000_H$ | W | 14, 13, 10-0 |
| | PPR0 | Port Pin Read register 0 | 16 | | $000C_H$ | $0000\ 0000_H$ | R | 14, 13, 10-0 |
| | PM0 | Port Mode register 0 | 16 | | $0010_H$ | $FBFF_H$ | R/W | 14, 13, 10-0 |
| | PMC0 | Port Mode Control register 0 | 16 | | $0014_H$ | $0000_H$ | R/W | 14, 13, 9-0 |
| | PFC0 | Port Function Control register 0 | 16 | | $0018_H$ | $0000_H$ | R/W | 14, 13, 9-0 |
| | PFCE0 | Port Function Control Expansion register 0 | 16 | | $001C_H$ | $0000_H$ | R/W | 14, 13, 9-0 |
| | PMSR0 | Port Mode Set Reset register 0 | 32 | | $0020_H$ | $0000\ FBFF_H$ | R/W | 30, 29, 26-16, 14, 13, 10-0 |
| | PMCSR0 | Port Mode Control Set Reset register 0 | 32 | | $0024_H$ | $0000\ 0000_H$ | R/W | 30, 29, 26-16, 14, 13, 9-0 |
| | PINV0 | Port output value Inversion register 0 | 16 | | $0030_H$ | $0000_H$ | R/W | 14, 13, 10-0 |
| | PIBC0 | Port Port Input Buffer Control register 0 | 16 | | $4000_H$ | $0000_H$ | R/W | 14, 13, 10-0 |
| | PBDC0 | Port Bi-Direction Control register 0 | 16 | | $4004_H$ | $0000_H$ | R/W | 14, 13, 10-0 |
| | PIPC0 | Port IP Control register 0 | 16 | | $4008_H$ | $0000_H$ | R/W | 13, 9-7, 5-0 |
| | PU0 | Pull-up option register 0 | 16 | | $400C_H$ | $0000_H$ | R/W | 14, 13, 10-0 |
| | PD0 | Pull-down option register 0 | 16 | | $4010_H$ | $0000_H$ | R/W | 14, 13, 10-0 |
| | PODC0 | Port Open Drain Control register 0 | 32 | | $4014_H$ | $0000\ 0400_H$ | R/W | 14, 13, 10-0 |
| | PDSC0 | Port Drive Strength Control register 0 | 32 | | $4018_H$ | $0000\ 0000_H$ | R/W | 14, 13, 10-0 |
| | PIS0 | Port Input Selection register 0 | 16 | | $401C_H$ | $0000_H$ | R/W | 14, 13, 10-0 |
| | PUCC0 | Port Universal Characteristics Control register 0 | 32 | | $4028_H$ | $0000\ 0000_H$ | R/W | 14, 13, 10-0 |
| | PODCE0 | Port Open Drain Control Expansion register 0 | 32 | | $4038_H$ | $0000\ 0000_H$ | R/W | 14, 13, 10-0 |

**Note:** Not all bit exist for all products. For details to the available port pins refer to **Table 2.1**

**Note:** RESET-OUT function.
The pin P0_10 has RESETOUTZ function emulated by port logic and special IO buffer. By this function, this pin can drive out low level during and after reset to reset external ASIC. This function is realized by having a special reset value of this pin being output mode by PM0_10 = 0 and being open-drain output buffer by PODC0_10 = 1, with the output value being low by P0_10 = 0. This function is effective for any kind of resets either it is external or internal.
Until being disabled by register settings, pin P0_10 keeps driving out low level after any kind of reset. To avoid data collision, the outside circuit connected to this pin must not drive in high level at any case.

## 2.4.4 Port 1 (P1)

### 2.4.4.1 Control Register

| Port | Register | Function | Access Size | Base Address | Offset Address | Reset Value | Existent Bit | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | Position |
| P1 | P1 | Port register 1 | 16 | FFC1 $0000_H$ | $0040_H$ | $0000_H$ | R/W | 7-1 |
| | PSR1 | Port Set Reset register 1 | 32 | | $0044_H$ | $0000\ 0000_H$ | R/W | 23-17, 7-1 |
| | PNOT1 | Port NOT register 1 | 16 | | $0048_H$ | $0000_H$ | W | 7-1 |
| | PPR1 | Port Pin Read register 1 | 16 | | $004C_H$ | $0000\ 0000_H$ | R | 7-1 |
| | PM1 | Port Mode register 1 | 16 | | $0050_H$ | $FFFF_H$ | R/W | 7-1 |
| | PMC1 | Port Mode Control register 1 | 16 | | $0054_H$ | $0000_H$ | R/W | 7-1 |
| | PFC1 | Port Function Control register 1 | 16 | | $0058_H$ | $0000_H$ | R/W | 7-1 |
| | PFCE1 | Port Function Control Expansion register 1 | 16 | | $005C_H$ | $0000_H$ | R/W | 7, 6, 4-1 |
| | PMSR1 | Port Mode Set Reset register 1 | 32 | | $0060_H$ | $0000\ FFFF_H$ | R/W | 23-17, 7-1 |
| | PMCSR1 | Port Mode Control Set Reset register 1 | 32 | | $0064_H$ | $0000\ 0000_H$ | R/W | 23-17, 7-1 |
| | PINV1 | Port output value Inversion register 1 | 16 | | $0070_H$ | $0000_H$ | R/W | 7-1 |
| | PIBC1 | Port Port Input Buffer Control register 1 | 16 | | $4040_H$ | $0000_H$ | R/W | 7-1 |
| | PBDC1 | Port Bi-Direction Control register 1 | 16 | | $4044_H$ | $0000_H$ | R/W | 7-1 |
| | PIPC1 | Port IP Control register 1 | 16 | | $4048_H$ | $0000_H$ | R/W | 4-1 |
| | PU1 | Pull-up option register 1 | 16 | | $404C_H$ | $0000_H$ | R/W | 7-1 |
| | PD1 | Pull-down option register 1 | 16 | | $4050_H$ | $0000_H$ | R/W | 7-1 |
| | PODC1 | Port Open Drain Control register 1 | 32 | | $4054_H$ | $0000\ 0000_H$ | R/W | 7-1 |
| | PDSC1 | Port Drive Strength Control register 1 | 32 | | $4058_H$ | $0000\ 0000_H$ | R/W | 7-1 |
| | PIS1 | Port Input Selection register 1 | 16 | | $405C_H$ | $0000_H$ | R/W | 7-1 |
| | PUCC1 | Port Universal Characteristics Control register 1 | 32 | | $4068_H$ | $0000\ 0000_H$ | R/W | 7-1 |
| | PODCE1 | Port Open Drain Control Expansion register 1 | 32 | | $4078_H$ | $0000\ 0000_H$ | R/W | 7-1 |

**Note:** Not all bit exist for all products. For details to the available port pins refer to **Table 2.1**

## 2.4.5 Port 2 (P2)

### 2.4.5.1 Control Register

| Port | Register | Function | Access Size | Base Address | Offset Address | Reset Value | Existent Bit | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | Position |
| P2 | P2 | Port register 2 | 16 | FFC1 $0000_H$ | $0080_H$ | $0000_H$ | R/W | 15-0 |
| | PSR2 | Port Set Reset register 2 | 32 | | $0084_H$ | 0000 $0000_H$ | R/W | 31-16, 15-0 |
| | PNOT2 | Port NOT register 2 | 16 | | $0088_H$ | $0000_H$ | W | 15-0 |
| | PPR2 | Port Pin Read register 2 | 16 | | $008C_H$ | 0000 $0000_H$ | R | 15-0 |
| | PM2 | Port Mode register 2 | 16 | | $0090_H$ | $FFFF_H$ | R/W | 15-0 |
| | PMC2 | Port Mode Control register 2 | 16 | | $0094_H$ | $0000_H$ | R/W | 15-0 |
| | PFC2 | Port Function Control register 2 | 16 | | $0098_H$ | $0000_H$ | R/W | 15-0 |
| | PFCE2 | Port Function Control Expansion register 2 | 16 | | $009C_H$ | $0000_H$ | R/W | 15-0 |
| | PMSR2 | Port Mode Set Reset register 2 | 32 | | $00A0_H$ | 0000 $FFFF_H$ | R/W | 31-16, 15-0 |
| | PMCSR2 | Port Mode Control Set Reset register 2 | 32 | | $00A4_H$ | 0000 $0000_H$ | R/W | 31-16, 15-0 |
| | PINV2 | Port output value Inversion register 2 | 16 | | $00B0_H$ | $0000_H$ | R/W | 15-0 |
| | PIBC2 | Port Port Input Buffer Control register 2 | 16 | | $4080_H$ | $0000_H$ | R/W | 15-0 |
| | PBDC2 | Port Bi-Direction Control register 2 | 16 | | $4084_H$ | $0000_H$ | R/W | 15-0 |
| | PIPC2 | Port IP Control register 2 | 16 | | $4088_H$ | $0000_H$ | R/W | 15-0 |
| | PU2 | Pull-up option register 2 | 16 | | $408C_H$ | $0000_H$ | R/W | 15-0 |
| | PD2 | Pull-down option register 2 | 16 | | $4090_H$ | $0000_H$ | R/W | 15-0 |
| | PODC2 | Port Open Drain Control register 2 | 32 | | $4094_H$ | 0000 $0000_H$ | R/W | 15-0 |
| | PDSC2 | Port Drive Strength Control register 2 | 32 | | $4098_H$ | 0000 $0000_H$ | R/W | 15-0 |
| | PIS2 | Port Input Selection register 2 | 16 | | $409C_H$ | $0000_H$ | R/W | 15-0 |
| | PUCC2 | Port Universal Characteristics Control register 2 | 32 | | $40A8_H$ | 0000 $0000_H$ | R/W | 15-0 |
| | PODCE2 | Port Open Drain Control Expansion register 2 | 32 | | 40B8H | 0000 $0000_H$ | R/W | 15-0 |

**Note:** Not all bit exist for all products. For details to the available port pins refer to **Table 2.1**

## 2.4.6 Port 3 (P3)

### 2.4.6.1 Control Register

| Port | Register | Function | Access Size | Base Address | Offset Address | Reset Value | Existent Bit | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | Position |
| P3 | P3 | Port register 3 | 16 | FFC1 $0000_H$ | $00C0_H$ | $0000_H$ | R/W | 14-0 |
| | PSR3 | Port Set Reset register 3 | 32 | | $00C4_H$ | 0000 $0000_H$ | R/W | 30-16, 14-0 |
| | PNOT3 | Port NOT register 3 | 16 | | $00C8_H$ | $0000_H$ | W | 14-0 |
| | PPR3 | Port Pin Read register 3 | 16 | | $00CC_H$ | 0000 $0000_H$ | R | 14-0 |
| | PM3 | Port Mode register 3 | 16 | | $00D0_H$ | $FFFF_H$ | R/W | 14-0 |
| | PMC3 | Port Mode Control register 3 | 16 | | $00D4_H$ | $0000_H$ | R/W | 14-0 |
| | PFC3 | Port Function Control register 3 | 16 | | $00D8_H$ | $0000_H$ | R/W | 14-0 |
| | PFCE3 | Port Function Control Expansion register 3 | 16 | | $00DC_H$ | $0000_H$ | R/W | 14-0 |
| | PMSR3 | Port Mode Set Reset register 3 | 32 | | $00E0_H$ | 0000 $FFFF_H$ | R/W | 30-16, 14-0 |
| | PMCSR3 | Port Mode Control Set Reset register 3 | 32 | | $00E4_H$ | 0000 $0000_H$ | R/W | 30-16, 14-0 |
| | PINV3 | Port output value Inversion register 3 | 16 | | $00F0_H$ | $0000_H$ | R/W | 14-0 |
| | PIBC3 | Port Port Input Buffer Control register 3 | 16 | | $40C0_H$ | $0000_H$ | R/W | 14-0 |
| | PBDC3 | Port Bi-Direction Control register 3 | 16 | | $40C4_H$ | $0000_H$ | R/W | 14-0 |
| | PIPC3 | Port IP Control register 3 | 16 | | $40C8_H$ | $0000_H$ | R/W | 14-8, 6-2 |
| | PU3 | Pull-up option register 3 | 16 | | $40CC_H$ | $0000_H$ | R/W | 14-0 |
| | PD3 | Pull-down option register 3 | 16 | | $40D0_H$ | $0000_H$ | R/W | 14-0 |
| | PODC3 | Port Open Drain Control register 3 | 32 | | $40D4_H$ | 0000 $0000_H$ | R/W | 14-0 |
| | PDSC3 | Port Drive Strength Control register 3 | 32 | | $40D8_H$ | 0000 $0000_H$ | R/W | 14-0 |
| | PIS3 | Port Input Selection register 3 | 16 | | $40DC_H$ | $0000_H$ | R/W | 14-0 |
| | PUCC3 | Port Universal Characteristics Control register 3 | 32 | | $40E8_H$ | 0000 $0000_H$ | R/W | 14-0 |
| | PODCE3 | Port Open Drain Control Expansion register 3 | 32 | | $40F8_H$ | 0000 $0000_H$ | R/W | 14-0 |

**Note:** Not all bit exist for all products. For details to the available port pins refer to **Table 2.1**

## 2.4.7 Port 4 (P4)

### 2.4.7.1 Control Register

| Port | Register | Function | Access Size | Base Address | Offset Address | Reset Value | Existent Bit | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | Position |
| P4 | P4 | Port register 4 | 16 | FFC1 $0000_H$ | $0100_H$ | $0000_H$ | R/W | 14-0 |
| | PSR4 | Port Set Reset register 4 | 32 | | $0104_H$ | 0000 $0000_H$ | R/W | 30-16, 14-0 |
| | PNOT4 | Port NOT register 4 | 16 | | $0108_H$ | $0000_H$ | W | 14-0 |
| | PPR4 | Port Pin Read register 4 | 16 | | $010C_H$ | 0000 $0000_H$ | R | 14-0 |
| | PM4 | Port Mode register 4 | 16 | | $0110_H$ | $FFFF_H$ | R/W | 14-0 |
| | PMC4 | Port Mode Control register 4 | 16 | | $0114_H$ | $0000_H$ | R/W | 14-0 |
| | PFC4 | Port Function Control register 4 | 16 | | $0118_H$ | $0000_H$ | R/W | 14-0 |
| | PFCE4 | Port Function Control Expansion register 4 | 16 | | $011C_H$ | $0000_H$ | R/W | 14-0 |
| | PMSR4 | Port Mode Set Reset register 4 | 32 | | $0120_H$ | 0000 $FFFF_H$ | R/W | 30-16, 14-0 |
| | PMCSR4 | Port Mode Control Set Reset register 4 | 32 | | $0124_H$ | 0000 $0000_H$ | R/W | 30-16, 14-0 |
| | PINV4 | Port output value Inversion register 4 | 16 | | $0130_H$ | $0000_H$ | R/W | 14-0 |
| | PIBC4 | Port Port Input Buffer Control register 4 | 16 | | $4100_H$ | $0000_H$ | R/W | 14-0 |
| | PBDC4 | Port Bi-Direction Control register 4 | 16 | | $4104_H$ | $0000_H$ | R/W | 14-0 |
| | PIPC4 | Port IP Control register 4 | 16 | | $4108_H$ | $0000_H$ | R/W | 14-10, 8, 5-0 |
| | PU4 | Pull-up option register 4 | 16 | | $410C_H$ | $0000_H$ | R/W | 14-0 |
| | PD4 | Pull-down option register 4 | 16 | | $4110_H$ | $0000_H$ | R/W | 14-0 |
| | PODC4 | Port Open Drain Control register 4 | 32 | | $4114_H$ | 0000 $0000_H$ | R/W | 14-0 |
| | PDSC4 | Port Drive Strength Control register 4 | 32 | | $4118_H$ | 0000_$0000_H$ | R/W | 14-0 |
| | PIS4 | Port Input Selection register 4 | 16 | | $411C_H$ | $0000_H$ | R/W | 14-0 |
| | PUCC4 | Port Universal Characteristics Control register 4 | 32 | | $4128_H$ | 0000 $0000_H$ | R/W | 14-0 |
| | PODCE4 | Port Open Drain Control Expansion register 4 | 32 | | $4138_H$ | 0000 $0000_H$ | R/W | 14-0 |

**Note:** Not all bit exist for all products. For details to the available port pins refer to **Table 2.1**

## 2.4.8 Port 5 (P5)

### 2.4.8.1 Control Register

| Port | Register | Function | Access Size | Base Address | Offset Address | Reset Value | Existent Bit | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | Position |
| P5 | P5 | Port register 5 | 16 | $FFC10000_H$ | $0140_H$ | $0000_H$ | R/W | 15-4, 1, 0 |
| | PSR5 | Port Set Reset register 5 | 32 | | $0144_H$ | 0000 $0000_H$ | R/W | 31-20, 17, 16, 15-4, 1, 0 |
| | PNOT5 | Port NOT register 5 | 16 | | $0148_H$ | $0000_H$ | W | 15-4, 1, 0 |
| | PPR5 | Port Pin Read register 5 | 16 | | $014C_H$ | 0000 $0000_H$ | R | 15-4, 1, 0 |
| | PM5 | Port Mode register 5 | 16 | | $0150_H$ | $FFFF_H$ | R/W | 15-4, 1, 0 |
| | PMC5 | Port Mode Control register 5 | 16 | | $0154_H$ | $0000_H$ | R/W | 15-4, 1, 0 |
| | PFC5 | Port Function Control register 5 | 16 | | $0158_H$ | $0000_H$ | R/W | 15-12, 10-4, 1, 0 |
| | PFCE5 | Port Function Control Expansion register 5 | 16 | | $015C_H$ | $0000_H$ | R/W | 15-12, 10-4, 1, 0 |
| | PMSR5 | Port Mode Set Reset register 5 | 32 | | $0160_H$ | 0000 $FFFF_H$ | R/W | 31-20, 17, 16, 15-4, 1, 0 |
| | PMCSR5 | Port Mode Control Set Reset register 5 | 32 | | $0164_H$ | 0000 $0000_H$ | R/W | 31-20, 17, 16, 15-4, 1, 0 |
| | PINV5 | Port output value Inversion register 5 | 16 | | $0170_H$ | $0000_H$ | R/W | 15-4, 1, 0 |
| | PIBC5 | Port Port Input Buffer Control register 5 | 16 | | $4140_H$ | $0000_H$ | R/W | 15-4, 1, 0 |
| | PBDC5 | Port Bi-Direction Control register 5 | 16 | | $4144_H$ | $0000_H$ | R/W | 15-4, 1, 0 |
| | PIPC5 | Port IP Control register 5 | 16 | | $4148_H$ | $0000_H$ | R/W | 15-6, 4, 1, 0 15-4, 1, 0 |
| | PU5 | Pull-up option register 5 | 16 | | $414C_H$ | $0000_H$ | R/W | 15-4, 1, 0 |
| | PD5 | Pull-down option register 5 | 16 | | $4150_H$ | $0000_H$ | R/W | 15-4, 1, 0 |
| | PODC5 | Port Open Drain Control register 5 | 32 | | $4154_H$ | 0000 $0000_H$ | R/W | 15-4, 1, 0 |
| | PDSC5 | Port Drive Strength Control register 5 | 32 | | $4158_H$ | 0000 $0000_H$ | R/W | 15-4, 1, 0 |
| | PIS5 | Port Input Selection register 5 | 16 | | $415C_H$ | $0000_H$ | R/W | 15-4, 1, 0 |
| | PUCC5 | Port Universal Characteristics Control register 5 | 32 | | $4168_H$ | 0000 $0000_H$ | R/W | 15-4, 1, 0 |
| | PODCE5 | Port Open Drain Control Expansion register 5 | 32 | | $4178_H$ | 0000 $0000_H$ | R/W | 15-4, 1, 0 |

**Note:** Not all bit exist for all products. For details to the available port pins refer to **Table 2.1**

## 2.4.9 Port 6 (P6)

### 2.4.9.1 Control Register

| Port | Register | Function | Access Size | Base Address | Offset Address | Reset Value | Existent Bit | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | Position |
| P6 | P6 | Port register 6 | 16 | $FFC10000_H$ | $0180_H$ | $0000_H$ | R/W | 15-0 |
| | PSR6 | Port Set Reset register 6 | 32 | | $0184_H$ | 0000 $0000_H$ | R/W | 31-16, 15-0 |
| | PNOT6 | Port NOT register 6 | 16 | | $0188_H$ | $0000_H$ | W | 15-0 |
| | PPR6 | Port Pin Read register 6 | 16 | | $018C_H$ | 0000 $0000_H$ | R | 15-0 |
| | PM6 | Port Mode register 6 | 16 | | $0190_H$ | $FFFF_H$ | R/W | 15-0 |
| | PMC6 | Port Mode Control register 6 | 16 | | $0194_H$ | 0000H | R/W | 15-0 |
| | PFC6 | Port Function Control register 6 | 16 | | $0198_H$ | 0000H | R/W | 15-0 |
| | PFCE6 | Port Function Control Expansion register 6 | 16 | | $019C_H$ | 0000H | R/W | 15-9, 7-0 |
| | PMSR6 | Port Mode Set Reset register 6 | 32 | | $01A0_H$ | 0000 $FFFF_H$ | R/W | 31-16, 15-0 |
| | PMCSR6 | Port Mode Control Set Reset register 6 | 32 | | $01A4_H$ | 0000 $0000_H$ | R/W | 31-16, 15-0 |
| | PINV6 | Port output value Inversion register 6 | 16 | | $01B0_H$ | $0000_H$ | R/W | 15-0 |
| | PIBC6 | Port Port Input Buffer Control register 6 | 16 | | $4180_H$ | $0000_H$ | R/W | 15-0 |
| | PBDC6 | Port Bi-Direction Control register 6 | 16 | | $4184_H$ | $0000_H$ | R/W | 15-0 |
| | PIPC6 | Port IP Control register 6 | 16 | | $4188_H$ | $0000_H$ | R/W | 15-3, 1, 0 |
| | PU6 | Pull-up option register 6 | 16 | | $418C_H$ | $0000_H$ | R/W | 15-0 |
| | PD6 | Pull-down option register 6 | 16 | | $4190_H$ | $0000_H$ | R/W | 15-0 |
| | PODC6 | Port Open Drain Control register 6 | 32 | | $4194_H$ | 0000 $0000_H$ | R/W | 15-0 |
| | PDSC6 | Port Drive Strength Control register 6 | 32 | | $4198_H$ | 0000 $0000_H$ | R/W | 15-0 |
| | PIS6 | Port Input Selection register 6 | 16 | | $419C_H$ | $0000_H$ | R/W | 15-0 |
| | PUCC6 | Port Universal Characteristics Control register 6 | 32 | | $41A8_H$ | 0000 $0000_H$ | R/W | 15-0 |
| | PODCE6 | Port Open Drain Control Expansion register 6 | 32 | | $41B8_H$ | 0000 $0000_H$ | R/W | 15-0 |

**Note:** Not all bit exist for all products. For details to the available port pins refer to **Table 2.1**

## 2.4.10 Port 7 (P7)

### 2.4.10.1 Control Register

| Port | Register | Function | Access Size | Base Address | Offset Address | Reset Value | Existent Bit | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | Position |
| P7 | P7 | Port register 7 | 16 | FFC1 $0000_H$ | $01C0_H$ | $0000_H$ | R/W | 9-0 |
| | PSR7 | Port Set Reset register 7 | 32 | | $01C4_H$ | 0000 $0000_H$ | R/W | 25-16, 9-0 |
| | PNOT7 | Port NOT register 7 | 16 | | $01C8_H$ | $0000_H$ | W | 9-0 |
| | PPR7 | Port Pin Read register 7 | 16 | | $01CC_H$ | 0000 $0000_H$ | R | 9-0 |
| | PM7 | Port Mode register 7 | 16 | | $01D0_H$ | $FFFF_H$ | R/W | 9-0 |
| | PMC7 | Port Mode Control register 7 | 16 | | $01D4_H$ | $0000_H$ | R/W | 9-0 |
| | PFC7 | Port Function Control register 7 | 16 | | $01D8_H$ | $0000_H$ | R/W | 9-0 |
| | PFCE7 | Port Function Control Expansion register 7 | 16 | | $01DC_H$ | $0000_H$ | R/W | 9-0 |
| | PMSR7 | Port Mode Set Reset register 7 | 32 | | $01E0_H$ | 0000 $FFFF_H$ | R/W | 25-16, 9-0 |
| | PMCSR7 | Port Mode Control Set Reset register 7 | 32 | | $01E4_H$ | 0000 $0000_H$ | R/W | 25-16, 9-0 |
| | PINV7 | Port output value Inversion register 7 | 16 | | $01F0_H$ | $0000_H$ | R/W | 9-0 |
| | PIBC7 | Port Port Input Buffer Control register 7 | 16 | | $41C0_H$ | $0000_H$ | R/W | 9-0 |
| | PBDC7 | Port Bi-Direction Control register 7 | 16 | | $41C4_H$ | $0000_H$ | R/W | 9-0 |
| | PIPC7 | Port IP Control register 7 | 16 | | $41C8_H$ | $0000_H$ | R/W | 3-0 |
| | PU7 | Pull-up option register 7 | 16 | | $41CC_H$ | $0000_H$ | R/W | 9-0 |
| | PD7 | Pull-down option register 7 | 16 | | $41D0_H$ | $0000_H$ | R/W | 9-0 |
| | PODC7 | Port Open Drain Control register 7 | 32 | | $41D4_H$ | $0000_0000_H$ | R/W | 9-0 |
| | PDSC7 | Port Drive Strength Control register 7 | 32 | | $41D8_H$ | $0000_0000_H$ | R/W | 9-0 |
| | PIS7 | Port Input Selection register 7 | 16 | | $41DC_H$ | $0000_H$ | R/W | 9-0 |
| | PUCC7 | Port Universal Characteristics Control register 7 | 32 | | $41E8_H$ | 0000 $0000_H$ | R/W | 9-0 |
| | PODCE7 | Port Open Drain Control Expansion register 7 | 32 | | $41F8_H$ | 0000 $0000_H$ | R/W | 9-0 |

**Note:** Not all bit exist for all products. For details to the available port pins refer to **Table 2.1**

## 2.4.11 Port 8 (P8)

### 2.4.11.1 Control Register

| Port | Register | Function | Access Size | Base Address | Offset Address | Reset Value | Existent Bit | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | Position |
| P8 | P8 | Port register 8 | 16 | FFC1 $0000_H$ | $0200_H$ | $0000_H$ | R/W | 15-0 |
| | PSR8 | Port Set Reset register 8 | 32 | | $0204_H$ | 0000 $0000_H$ | R/W | 31-16, 15-0 |
| | PNOT8 | Port NOT register 8 | 16 | | $0208_H$ | $0000_H$ | W | 15-0 |
| | PPR8 | Port Pin Read register 8 | 16 | | $020C_H$ | 0000 $0000_H$ | R | 15-0 |
| | PM8 | Port Mode register 8 | 16 | | $0210_H$ | $FFFF_H$ | R/W | 15-0 |
| | PMC8 | Port Mode Control register 8 | 16 | | $0214_H$ | $0000_H$ | R/W | 15-0 |
| | PFC8 | Port Function Control register 8 | 16 | | $0218_H$ | $0000_H$ | R/W | 15-0 |
| | PFCE8 | Port Function Control Expansion register 8 | 16 | | $021C_H$ | $0000_H$ | R/W | 15-0 |
| | PMSR8 | Port Mode Set Reset register 8 | 32 | | $0220_H$ | 0000 $FFFF_H$ | R/W | 31-16, 15-0 |
| | PMCSR8 | Port Mode Control Set Reset register 8 | 32 | | $0224_H$ | 0000 $0000_H$ | R/W | 31-16, 15-0 |
| | PINV8 | Port output value Inversion register 8 | 16 | | $0230_H$ | $0000_H$ | R/W | 15-0 |
| | PIBC8 | Port Port Input Buffer Control register 8 | 16 | | $4200_H$ | $0000_H$ | R/W | 15-0 |
| | PBDC8 | Port Bi-Direction Control register 8 | 16 | | $4204_H$ | $0000_H$ | R/W | 15-0 |
| | PIPC8 | Port IP Control register 8 | 16 | | $4208_H$ | $0000_H$ | R/W | 15-8, 3-0 |
| | PU8 | Pull-up option register 8 | 16 | | $420C_H$ | $0000_H$ | R/W | 15-0 |
| | PD8 | Pull-down option register 8 | 16 | | $4210_H$ | $0000_H$ | R/W | 15-0 |
| | PODC8 | Port Open Drain Control register 8 | 32 | | $4214_H$ | 0000 $0000_H$ | R/W | 15-0 |
| | PDSC8 | Port Drive Strength Control register 8 | 32 | | $4218_H$ | 0000 $0000_H$ | R/W | 15-0 |
| | PIS8 | Port Input Selection register 8 | 16 | | $421C_H$ | $0000_H$ | R/W | 15-0 |
| | PUCC8 | Port Universal Characteristics Control register 8 | 32 | | $4228_H$ | 0000 $0000_H$ | R/W | 15-0 |
| | PODCE8 | Port Open Drain Control Expansion register 8 | 32 | | $4238_H$ | 0000 $0000_H$ | R/W | 15-0 |

**Note:** Not all bit exist for all products. For details to the available port pins refer to **Table 2.1**

## 2.4.12 Port 9 (P9)

### 2.4.12.1 Control Register

| Port | Register | Function | Access Size | Base Address | Offset Address | Reset Value | Existent Bit | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | R/W | Position |
| P9 | P9 | Port register 9 | 16 | $FFC10000_H$ | $0240_H$ | $0000_H$ | R/W | 8-0 |
| | PSR9 | Port Set Reset register 9 | 32 | | $0244_H$ | 0000 $0000_H$ | R/W | 24-16, 8-0 |
| | PNOT9 | Port NOT register 9 | 16 | | $0248_H$ | $0000_H$ | W | 8-0 |
| | PPR9 | Port Pin Read register 9 | 16 | | $024C_H$ | 0000 $0000_H$ | R | 8-0 |
| | PM9 | Port Mode register 9 | 16 | | $0250_H$ | $FFFF_H$ | R/W | 8-0 |
| | PMC9 | Port Mode Control register 9 | 16 | | $0254_H$ | $0000_H$ | R/W | 8-0 |
| | PFC9 | Port Function Control register 9 | 16 | | $0258_H$ | $0000_H$ | R/W | 8-5, 3-0 |
| | PFCE9 | Port Function Control Expansion register 9 | 16 | | $025C_H$ | $0000_H$ | R/W | 6, 5, 3-0 |
| | PMSR9 | Port Mode Set Reset register 9 | 32 | | $0260_H$ | 0000 $FFFF_H$ | R/W | 24-16, 8-0 |
| | PMCSR9 | Port Mode Control Set Reset register 9 | 32 | | $0264_H$ | 0000 $0000_H$ | R/W | 24-16, 8-0 |
| | PINV9 | Port output value Inversion register 9 | 16 | | $0270_H$ | $0000_H$ | R/W | 8-0 |
| | PIBC9 | Port Port Input Buffer Control register 9 | 16 | | $4240_H$ | $0000_H$ | R/W | 8-0 |
| | PBDC9 | Port Bi-Direction Control register 9 | 16 | | $4244_H$ | $0000_H$ | R/W | 8-0 |
| | PIPC9 | Port IP Control register 9 | 16 | | $4248_H$ | $0000_H$ | R/W | 8-1 |
| | PU9 | Pull-up option register 9 | 16 | | $424C_H$ | $0000_H$ | R/W | 8-0 |
| | PD9 | Pull-down option register 9 | 16 | | $4250_H$ | $0000_H$ | R/W | 8-0 |
| | PODC9 | Port Open Drain Control register 9 | 32 | | $4254_H$ | 0000 $0000_H$ | R/W | 8-0 |
| | PDSC9 | Port Drive Strength Control register 9 | 32 | | $4258_H$ | 0000 $0000_H$ | R/W | 8-0 |
| | PIS9 | Port Input Selection register 9 | 16 | | $425C_H$ | $0000_H$ | R/W | 8-0 |
| | PUCC9 | Port Universal Characteristics Control register 9 | 32 | | $4268_H$ | 0000 $0000_H$ | R/W | 8-0 |
| | PODCE9 | Port Open Drain Control Expansion register 9 | 32 | | $4278_H$ | 0000 $0000_H$ | R/W | 8-0 |

**Note:** Not all bit exist for all products. For details to the available port pins refer to **Table 2.1**

## 2.4.13 Register Address Mapping

### 2.4.13.1 Offset Address

To make a port macro reusable over different products, the offset address decoder in port macro is also standardized as the following tables

**Table 2.9 Offset Address of Numeric Port Register**

| | | Numeric Port Category (64 ports available) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | User Register Type (16 registers available. 16KB) | | | | | OS Register Type (16 registers available. 16KB) | | |
| | | P | PSR | PNOT | … | ..... | PIBC | … | … |
| 64 ports | Port 0 | 0000 | 0004 | 0008 | … | 003C | 4000 | … | 403C |
| | Port 1 | 0040 | 0044 | 0048 | … | 007C | 4040 | … | 407C |
| | … | … | … | ….. | … | … | … | … | … |
| | Port 63 | 0FC0 | 0FC4 | 0FC8 | … | 0FFC | 4FC0 | … | 4FFC |
| RFU (LowerByte) | Port 0 | 1000 | … | … | … | 103C | 5000 | … | 503C |
| | … | … | … | … | … | … | … | … | … |
| | Port 63 | 1FC0 | … | … | … | 1FFC | 5FC0 | … | 5FFC |
| RFU (UpperByte) | Port 0 | 2000 | … | … | … | 203C | 6000 | … | 603C |
| | … | … | … | … | … | … | … | … | … |
| | Port 63 | 2FC0 | … | … | … | 2FFC | 6FC0 | … | 6FFC |
| RFU | Port 0 | 3000 | … | … | … | 303C | 7000 | … | 703C |
| | … | … | … | … | … | … | … | … | … |
| | Port 64 | 3FC0 | … | … | … | 3FFC | 7FC0 | … | 7FFC |

Port registers were grouped into two types, User register type and OS register type.

For protection both types are considered as one port instance for the P-Bus Guard.

For more details about P-Bus guard, see **Section 3, CPU System**.

**Table 2.10 Offset Address of JTAG Port Register**

| | | JTAG Port Category (4 ports available though only 1 is defined as JTAG port) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | User Register Type (16 registers available. 1KB) | | | | | OS Register Type (16 registers available. 1KB) | | |
| | | JP | JPSR | JPNOT | ... | ... | JPIBC | ... | ... |
| | JPort0 | 0000 | 0004 | 0008 | | 003C | 4000 | | 403C |
| RFU | JPort1 | 0040 | 0044 | 0048 | ... | 007C | 4040 | ... | 407C |
| | JPort2 | 0080 | 0084 | 0088 | ... | 00BC | 4080 | ... | 408C |
| | JPort3 | 00C0 | 00C4 | 00C8 | | 00FC | 40C0 | | 40FC |
| | JPort0 | 1000 | | | | 103C | 5000 | | 503C |
| RFU (LowerByte) | ... | ... | ... | | ... | ... | ... | ... | ... |
| | JPort3 | 10C0 | | | | 10FC | 50C0 | | 50FC |
| | JPort0 | 2000 | | | | 203C | 6000 | | 603C |
| RFU (UpperByte) | ... | ... | ... | | ... | ... | ... | ... | ... |
| | JPort3 | 20C0 | | | | 20FC | 60C0 | | 60FC |
| | JPort0 | 3000 | | | | 303C | 7000 | | 703C |
| RFU | ... | ... | ... | | ... | ... | ... | ... | ... |
| | JPort3 | 30C0 | | | | 30FC | 70C0 | | 70FC |

## 2.5 Port Register Description

The list of port control register types are described below.

Table 2.11 Port Register Overview

| Port Register Name | Port Symbol | Offset Address | |
|---|---|---|---|
| | | Numeric Port | JTAG Port |
| Port register | P | 0x0000 + 0x40*n | 0x0000 + 0x40*n |
| Port Set Reset register | PSR | 0x0004 + 0x40*n | 0x0004 + 0x40*n |
| Port NOT register | PNOT | 0x0008 + 0x40*n | 0x0008 + 0x40*n |
| Port Pin Read register | PPR | 0x000C + 0x40*n | 0x000C + 0x40*n |
| Port Mode register | PM | 0x0010 + 0x40*n | 0x0010 + 0x40*n |
| Port Mode Control register | PMC | 0x0014 + 0x40*n | 0x0014 + 0x40*n |
| Port Function Control register | PFC | 0x0018 + 0x40*n | 0x0018 + 0x40*n |
| Port Function Control Expansion register | PFCE | 0x001C + 0x40*n | 0x001C + 0x40*n |
| Port Mode Set Reset register | PMSR | 0x0020 + 0x40*n | 0x0020 + 0x40*n |
| Port Mode Control Set Reset register | PMCSR | 0x0024 + 0x40*n | 0x0024 + 0x40*n |
| Port output value Inversion register | PINV | 0x0030 + 0x40*n | 0x0030 + 0x40*n |
| Port Input Buffer Control register | PIBC | 0x4000 + 0x40*n | 0x4000 + 0x40*n |
| Port Bi-Direction Control register | PBDC | 0x4004 + 0x40*n | 0x4004 + 0x40*n |
| Port IP Control register | PIPC | 0x4008 + 0x40*n | 0x4008 + 0x40*n |
| Pull-Up option register | PU | 0x400C + 0x40*n | 0x400C + 0x40*n |
| Pull-Down option register | PD | 0x4010 + 0x40*n | 0x4010 + 0x40*n |
| Port Open Drain Control register | PODC | 0x4014 + 0x40*n | 0x4014 + 0x40*n |
| Port Drive Strength Control register | PDSC | 0x4018 + 0x40*n | 0x4018 + 0x40*n |
| Port Input buffer Selection register | PIS | 0x401C + 0x40*n | 0x401C + 0x40*n |
| Port Universal Characteristic Control register | PUCC | 0x4028 + 0x40*n | 0x4028 + 0x40*n |
| Port Open Drain Control Expansion register | PODCE | 0x4038 + 0x40*n | 0x4038 + 0x40*n |
| Port Control register | PCR | 0x2000 + 0x40*n + 0x4*m | 0x2000 + 0x4*m |

**Note:** n = Port group numbers, m = Bit numbers of Port

**CAUTION**

**To prevent malfunction, secure reset value to registers that are not available for the individual product. For port availability of each product see Table 2.1.**

## 2.5.1 Pn/JP0 — Port Register

This register defines port pins output levels for port output mode.

**Access:** Pn: This register can be read / written in 16-bit units.
JP0: This register can be read / written in 8-bit units.

**Address:** Pn: <PORTn_base> + $0000_H$ + n × $40_H$
JP0: < JPORT0_base> + $0000_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Pn_15 | Pn_14 | Pn_13 | Pn_12 | Pn_11 | Pn_10 | Pn_9 | Pn_8 | Pn_7 | Pn_6 | Pn_5 | Pn_4 | Pn_3 | Pn_2 | Pn_1 | Pn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.12 Pn/JP0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | Pn_[15:0] | Sets the output level of pin Pn_m (m = 0 to 15).<br>0: Port pin drives low level<br>1: Port pin drives high level |

**NOTES**

1. Reading Pn returns the register value independent from other register settings.
2. The value on this register bit is reflected to a pin level in the following conditions.
   Case : Port Mode (PMCn_m = 0) & Output Mode (PMn_m = 0)

## 2.5.2 PPRn/JPPR0 — Port Pin Read Register

This register reflects the actual pin level when the input buffer is active.

**Access:** PPRn: This register can be read only in 16-bit units.
JPPR0: This register can be read only in 8-bit units.
This register is read only. Writing this register is ignored and the written data is discarded

**Address:** PPRn: <PORTn_base> + $000C_H$ + n × $40_H$
JPPR0: < JPORT0_base> + $000C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PPRn_15 | PPRn_14 | PPRn_13 | PPRn_12 | PPRn_11 | PPRn_10 | PPRn_9 | PPRn_8 | PPRn_7 | PPRn_6 | PPRn_5 | PPRn_4 | PPRn_3 | PPRn_2 | PPRn_1 | PPRn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 2.13 PPRn/JPPR0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PPRn_[15:0] | This register reflects the actual pin level when the input buffer is active<br>0: Port pin is at low level<br>1: Port pin is at high level |

### 2.5.3 PMn/JPM0 — Port Mode Register

This register selects the pin direction as input or output.

**Access:** PMn: This register can be read/written in 16-bit units.
JPM0: This register can be read/written in 8-bit units.

**Address:** PMn: <PORTn_base> + $0010_H$ + n × $40_H$
JPM0: < JPORT0_base> + $0010_H$

**Value after reset:** $FFFF_H$*1

Note 1. The PM0 register is $FBFF_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMn_15 | PMn_14 | PMn_13 | PMn_12 | PMn_11 | PMn_10 | PMn_9 | PMn_8 | PMn_7 | PMn_6 | PMn_5 | PMn_4 | PMn_3 | PMn_2 | PMn_1 | PMn_0 |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.14 PMn/JPM0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PMn_[15:0] | Specifies input/output mode of the corresponding pin.<br>0: Output mode (output enabled)<br>1: Input mode (output disabled) |

## 2.5.4 PMCn/JPMC0 — Port Mode Control Register

This register specifies whether the individual pins of port group n are in port mode or in alternative mode.

**Access:** PMCn: This register can be read/written in 16-bit units.
JPMC0: This register can be read/written in 8-bit units.

**Address:** PMCn: <PORTn_base> + $0014_H$ + n × $40_H$
JPMC0: <JPORT0_base> + $0014_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCn_15 | PMCn_14 | PMCn_13 | PMCn_12 | PMCn_11 | PMCn_10 | PMCn_9 | PMCn_8 | PMCn_7 | PMCn_6 | PMCn_5 | PMCn_4 | PMCn_3 | PMCn_2 | PMCn_1 | PMCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.15 PMCn/JPMC0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PMCn_[15:0] | Specifies the operation mode of the corresponding pin.<br>0: Port mode<br>1: Alternative mode |

## 2.5.5 PFCn — Port Function Control Register

This register selects the alternative peripheral functions together with PMn in Control Mode (PMCn = 1)

**Access:** PFCn: This register can be read/written in 16-bit units.

**Address:** PFCn: <PORTn_base> + $0018_H$ + n × $40_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFCn_15 | PFCn_14 | PFCn_13 | PFCn_12 | PFCn_11 | PFCn_10 | PFCn_9 | PFCn_8 | PFCn_7 | PFCn_6 | PFCn_5 | PFCn_4 | PFCn_3 | PFCn_2 | PFCn_1 | PFCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.16 PFCn register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PFCn_[15:0] | Specifies the alternative function of a pin.<br>See **Table 2.17, Setting alternative functions** |

**Table 2.17 Setting alternative functions**

| PFCn_[15:0] | PMn_[15:0] | Function |
|---|---|---|
| 0 | 1 | Alternative peripheral function 1 (Control Mode 1) Input |
| | 0 | Alternative peripheral function 1 (Control Mode 1) Output |
| 1 | 1 | Alternative peripheral function 2 (Control Mode 2) Input |
| | 0 | Alternative peripheral function 2 (Control Mode 2) Output |

**Note:** Do not set the control mode at the same time in the share function pins.

## 2.5.6 PFCEn/JPFCE0 — Port Function Control Expansion Register

This register selects the alternative peripheral functions together with PFCn and PMn in Control Mode (PMCn = 1).

**Access:** PFCEn: This register can be read/written in 16-bit units.
JPFCE0: This register can be read/written in 8-bit units.

**Address:** PFCEn: <PORTn_base> + $001C_H$ + n × $40_H$
JPFCE0 : <JPORT0_base> + $001C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFCEn_15 | PFCEn_14 | PFCEn_13 | PFCEn_12 | PFCEn_11 | PFCEn_10 | PFCEn_9 | PFCEn_8 | PFCEn_7 | PFCEn_6 | PFCEn_5 | PFCEn_4 | PFCEn_3 | PFCEn_2 | PFCEn_1 | PFCEn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.18 PFCEn/JPFCE0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PFCEn_[15:0] | Specifies the alternative function of a pin.<br>See **Table 2.19, Setting alternative functions.** |

**Table 2.19 Setting alternative functions**

| PFCEn_m | PFCn_m | PMn_m | Function |
|---|---|---|---|
| 0 | 0 | 1 | Alternative peripheral function 1 (Control Mode 1) Input |
| | | 0 | Alternative peripheral function 1 (Control Mode 1) Output |
| | 1 | 1 | Alternative peripheral function 2 (Control Mode 2) Input |
| | | 0 | Alternative peripheral function 2 (Control Mode 2) Output |
| 1 | 0 | 1 | Alternative peripheral function 3 (Control Mode 3) Input |
| | | 0 | Alternative peripheral function 3 (Control Mode 3) Output |
| | 1 | 1 | Alternative peripheral function 4 (Control Mode 4) Input |
| | | 0 | Alternative peripheral function 4 (Control Mode 4) Output |

Note 1. Output functions can possibly be assigned to more than one port for parallel usage. The output timing can differ between the ports because it depends on buffer selection and pin connection.

Note 2. Only activate one single pin to one given alternative input function. Do not activate a input function on multiple pins at the same time.
[e.g.]
MTTCAN0RX is assigned to the following pins on this device. When the 2nd input alternative function of P5_15 is selected, using the 3rd input alternative function of P2_6 and the 3rd input alternative function of P4_5 are prohibited.
- P2_6(3rd input alternative function)
- P4_5(3rd input alternative function)
- P5_15(2nd input alternative function)

### 2.5.7 PNOTn/JPNOT0 — Port Not Register

This register provided a method to flip the bit values of Pn register. The bits of Pn register are flipped if the PNOTn register is written with the corresponding bit values being 1.

**Access:** PNOTn: This register can be read / written in 16-bit units. The value is always read as $0000_H$.
JPNOTn: This register can be read / written in 8-bit units. The value is always read as $00_H$.

**Address:** PNOTn: <PORTn_base> + $0008_H$ + n × $40_H$
JPNOT0: <JPORT0_base> + $0008_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PNOTn_15 | PNOTn_14 | PNOTn_13 | PNOTn_12 | PNOTn_11 | PNOTn_10 | PNOTn_9 | PNOTn_8 | PNOTn_7 | PNOTn_6 | PNOTn_5 | PNOTn_4 | PNOTn_3 | PNOTn_2 | PNOTn_1 | PNOTn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 2.20 PNOTn/JPNOT0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PNOTn_[15:0] | 0: No effect on the value of Pn_m bit.<br>1: The value of Pn_m bit is flipped |

**NOTE**

Only write operation on this register is relevant.

Reading PNOTn register always returns 0.

## 2.5.8 PSRn/JPSR0 — Port Set Reset Register

This register provides an alternative method to write/read data on Pn register.

The upper 16 bits of PSRn act as a mask which specifies whether or not the value Pn.Pn_m is set by the corresponding bit in the lower 16 bits of PSRn.

**Access:** PSRn: This register can be read / written in 32-bit units.

**Address:** PSRn: <PORTn_base> + $0004_H$ + n × $40_H$
JPSR0: <JPORT0_base> + $0004_H$

**Value after reset:** $0000\ 0000_H$

A reset from any source initialize the bits

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSRn_31 | PSRn_30 | PSRn_29 | PSRn_28 | PSRn_27 | PSRn_26 | PSRn_25 | PSRn_24 | PSRn_23 | PSRn_22 | PSRn_21 | PSRn_20 | PSRn_19 | PSRn_18 | PSRn_17 | PSRn_16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSRn_15 | PSRn_14 | PSRn_13 | PSRn_12 | PSRn_11 | PSRn_10 | PSRn_9 | PSRn_8 | PSRn_7 | PSRn_6 | PSRn_5 | PSRn_4 | PSRn_3 | PSRn_2 | PSRn_1 | PSRn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.21 PSRn/JPSR0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 31 to 16 | PSRn_[31:16] | Specifies whether the value of the corresponding lower bit PSRn_m value is written to Pn_m.<br>0: Pn_m is not affected by PSRn_m.<br>1: Pn_m is PSRn_m |
| 15 to 0 | PSRn_[15:0] | Sets the output level of pin Pn_m (m = 0 to 15).<br>0: Low level is written on Pn_m when it is enabled by PSRn_(m+16)<br>1: High level is written on Pn_m when it is enabled by PSRn_(m+16) |

**NOTE**

Reading PSRn_[31:16] bits always returns 0.

Reading PSRn_[15:0] returns the value of Pn_[15:0].

### 2.5.9 PMSRn/JPMSR0 — Port Mode Set/Reset Register

This register provides an alternative method to write/read data on PMn register.

The upper 16 bits of PMSRn act as a mask which specifies whether or not the value PMn.PMn_m is set by the corresponding bit in the lower 16 bits of PMSRn.

**Access:** This register can be read/written in 32-bit units. Bits 31 to 16 are always read as $0000_H$. Reading bits 15 to 0 returns the value of register PMn.

**Address:** PMSRn: <PORTn_base> + $0020_H$ + n × $40_H$
JPMSR0: <JPORT0_base> + $0020_H$

**Value after reset:** 0000 $FFFF_H$*1

Note 1. The PMSR0 register is 0000 $FBFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMSRn_31 | PMSRn_30 | PMSRn_29 | PMSRn_28 | PMSRn_27 | PMSRn_26 | PMSRn_25 | PMSRn_24 | PMSRn_23 | PMSRn_22 | PMSRn_21 | PMSRn_20 | PMSRn_19 | PMSRn_18 | PMSRn_17 | PMSRn_16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMSRn_15 | PMSRn_14 | PMSRn_13 | PMSRn_12 | PMSRn_11 | PMSRn_10 | PMSRn_9 | PMSRn_8 | PMSRn_7 | PMSRn_6 | PMSRn_5 | PMSRn_4 | PMSRn_3 | PMSRn_2 | PMSRn_1 | PMSRn_0 |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.22 PMSRn/JPMSR0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 31 to 16 | PMSRn_[31:16] | Specifies whether the value of the corresponding lower bit PMSRn_m value is written to PMn_m.<br>0: PMn_m is not affected by PMSRn_m.<br>1: PMn_m is PMSRn_m<br>Example:<br>If PMSRn.PMSRn_31 = 1, the value of bit PMSRn.PMSRn_15 is written to bit PMn.PMn_15. |
| 15 to 0 | PMSRn_[15:0] | Data bits that specify the PMn_m value if the corresponding upper bit (PMSRn_[31:16]) PMSRn_m is 1.<br>0: PMn_m = 0<br>1: PMn_m = 1 |

**NOTE**

Reading PMSRn_[15:0] returns the value of PMn_[15:0].

## 2.5.10 PMCSRn/JPMCSR0 — Port Mode Control Set/Reset Register

This register provides an alternative method to write data to the PMCn register.

The upper 16 bits of PMCSRn act as a mask which specifies whether or not the value PMCn.PMCn_m is set by the corresponding bit in the lower 16 bits of PMCSRn.

**Access:** This register can be read/written in 32-bit units. Bits 31 to 16 are always read as $0000_H$. Reading bits 15 to 0 returns the value of register PMCn.

**Address:** PMCSRn: <PORTn_base> + $0024_H$ + n × $40_H$
JPMCSR0: <JPORT0_base> + $0024_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCSRn_31 | PMCSRn_30 | PMCSRn_29 | PMCSRn_28 | PMCSRn_27 | PMCSRn_26 | PMCSRn_25 | PMCSRn_24 | PMCSRn_23 | PMCSRn_22 | PMCSRn_21 | PMCSRn_20 | PMCSRn_19 | PMCSRn_18 | PMCSRn_17 | PMCSRn_16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCSRn_15 | PMCSRn_14 | PMCSRn_13 | PMCSRn_12 | PMCSRn_11 | PMCSRn_10 | PMCSRn_9 | PMCSRn_8 | PMCSRn_7 | PMCSRn_6 | PMCSRn_5 | PMCSRn_4 | PMCSRn_3 | PMCSRn_2 | PMCSRn_1 | PMCSRn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.23 PMCSRn/JPMCSR0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 31 to 16 | PMCSRn_ [31:16] | Specifies whether the value of the corresponding lower bit PMCSRn_m value is written to PMCn_m.<br>0: PMCn_m is not affected by PMCSRn_m.<br>1: PMCn_m is PMCSRn_m<br>Example:<br>If PMCSRn.PMCSRn_31 = 1, the value of bit PMCSRn.PMCSRn15 is written to bit PMCn.PMCn15. |
| 15 to 0 | PMCSRn_[15:0] | Data bits that specify the PMCn_m value if the corresponding upper bit (PMCSRn_[31:16]) PMCSRn_m is 1.<br>0: PMCn_m = 0<br>1: PMCn_m = 1 |

## 2.5.11 PINVn/JPINV0 — Port output value Inversion Register

This register inverts the output value of the port.

**Access:** PINVn: This register can be read/written in 16-bit units.
JPINV0: This register can be read/written in 8-bit units.

**Address:** PINVn: <PORTn_base> + $0030_H$ + n × $40_H$
JPINV0: <JPORT0_base> + $0030_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PINVn_15 | PINVn_14 | PINVn_13 | PINVn_12 | PINVn_11 | PINVn_10 | PINVn_9 | PINVn_8 | PINVn_7 | PINVn_6 | PINVn_5 | PINVn_4 | PINVn_3 | PINVn_2 | PINVn_1 | PINVn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.24 PINVn/JPINV0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PINVn_[15:0] | This register inverts the output value of the port.<br>0: No effect<br>1: Inverted value is output |

## 2.5.12 PIBCn/JPIBC0 — Port Input Buffer Control Register

This register is used as one of the factors to enable/disable port pin's input buffer in Port Mode (PMC = 0).

**Access:** PIBCn: This register can be read/written in 16-bit units.
JPIBC0: This register can be read/written in 8-bit units.

**Address:** PIBCn: <PORTn_base> + $4000_H$ + n × $40_H$
JPIBC0: <JPORT0_base> + $4000_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PIBCn_15 | PIBCn_14 | PIBCn_13 | PIBCn_12 | PIBCn_11 | PIBCn_10 | PIBCn_9 | PIBCn_8 | PIBCn_7 | PIBCn_6 | PIBCn_5 | PIBCn_4 | PIBCn_3 | PIBCn_2 | PIBCn_1 | PIBCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.25 PIBCn/JPIBC0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PIBCn_[15:0] | Enables/disables the input buffer.<br>0: Input buffer disabled<br>1: Input buffer enabled. |

**NOTES**

1. To enable port pin's input buffer, the IO direction must be set as input mode by PM = 1 during Port Mode (PMC = 0).
2. By keeping this register at a reset value of 0, port pin's input buffer does not consumes current even when the pin level is at an intermediate voltage. When the input buffer is disabled, in port mode, through current does not flow even when the pin level is Hi-Z. Thus the pin does not need to be fixed to a high or low level externally.
3. During this register set 1 with Port Mode (PMC = 0), the values for peripheral macro are fixed.
4. During "Software I/O control altermative-function input" Mode (PMC = 1, PM = 1, PIPC = 0), this register bit must be set to 0.

## 2.5.13 PBDCn/JPBDC0 — Port Bi-Direction Control Register

This register enables the input buffer and sets the port to bi-directional mode. In bi-direction mode, PPRn.PPRn_m can read the level of the Pn_m pin.

**Access:** PBDCn: This register can be read/written in 16-bit units.
JPBDC0: This register can be read/written in 8-bit units.

**Address:** PBDCn: <PORTn_base> + $4004_H$ + n × $40_H$
JPBDC0: <JPORT0_base> + $4004_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PBDCn_15 | PBDCn_14 | PBDCn_13 | PBDCn_12 | PBDCn_11 | PBDCn_10 | PBDCn_9 | PBDCn_8 | PBDCn_7 | PBDCn_6 | PBDCn_5 | PBDCn_4 | PBDCn_3 | PBDCn_2 | PBDCn_1 | PBDCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.26 PBDCn/JPBDC0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PBDCn_[15:0] | Enables/disables bi-directional mode of the corresponding pin.<br>0: Bi-directional mode disabled<br>1: Bi-directional mode enabled |

**NOTE**

Loopback is enabled after four cycles of CLK_LSB since this register is wrriten.

## 2.5.14 PIPCn — Port IP Control Register

This register specifies whether the I/O direction of pin Pn_m is controlled by the port mode register PMn.PMn_m or by an alternative function. If pin Pn_m is operated in alternative mode (PMCn.PMCn_m = 1) and the alternative function requires direct control of the I/O direction, then PIPCn.PIPCn_m must be set to 1 as well. This transfers I/O control to the alternative function and overrules the PMn.PMn_m setting. The list of alternative function that require direct control of the I/O direction is provided **Table 2.28**. PIPC must be set to 1 when these alternative function is selected.

**Access:** PIPCn: This register can be read/written in 16-bit units.

**Address:** PIPCn: <PORTn_base> + $4008_H$ + n × $40_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PIPCn_15 | PIPCn_14 | PIPCn_13 | PIPCn_12 | PIPCn_11 | PIPCn_10 | PIPCn_9 | PIPCn_8 | PIPCn_7 | PIPCn_6 | PIPCn_5 | PIPCn_4 | PIPCn_3 | PIPCn_2 | PIPCn_1 | PIPCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.27 PIPCn register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PIPCn_[15:0] | Specifies the I/O control mode.<br>0: I/O mode is selected by PMn.PMn_m (software I/O control).<br>1: I/O mode is selected by the peripheral function (direct I/O control). |

**Table 2.28 Direct Control of the I/O direction by Alternative Function**

| Function | | | Explanation | Note |
|---|---|---|---|---|
| Group | Name | I/O | | |
| GTM | GTMATnOm | O | GTM ATOM output | n=0-2, m=0-3, x=0-3 |
| | GTMATnOxN | O | | |
| CSIH | CSIHnDCSm | I | CSIH serial data consequency signal | n=0-3, m=0-2 |
| | CSIHnSOm | O | CSIH serial data output | |
| | CSIHnSCIm | I | CSIH serial clock input signal | |
| | CSIHnSCOm | O | CSIH serial clock output signal | |
| | CSIHnRYI | I | CSIH ready (1) / busy (0) input signal | |
| | CSIHnRYO | O | CSIH ready (1) / busy (0) output signal | |
| HS-USRT | HSURTnSDIOmI | I | HSURT data input | n=0-3 |
| | HSURTnSDIOmO | O | HSURT data output | |
| | HSURTnSCKI | I | HSURT serial clock input | |
| | HSURTnSCKO | O | HSURT serial clock output | |
| | HSURTnCSI | I | HSURT chip select input | |
| | HSURTnCSO | O | HSURT chip select output | |
| MEMC | MEMCnDmI | I | External memory interface data bus input | n=0, m=0-7 |
| | MEMCnDmO | O | External memory interface data bus output | |
| ETNA | ETHnMDI | I | ETH Serial management interface data Input | n=0-1 |
| | ETHnMDO | O | ETH Serial management interface data Output | |

## 2.5.15 PUn/JPU0 — Pull-Up Option Register

This register specifies whether pull-up resistor is connected to an input pin.

**Access:** PUn: This register can be read/written in 16-bit units.
JPU0: This register can be read/written in 8-bit units.

**Address:** PUn: <PORTn_base> + $400C_H$ + n × $40_H$
JPU0: <JPORT0_base> + $400C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PUn_15 | PUn_14 | PUn_13 | PUn_12 | PUn_11 | PUn_10 | PUn_9 | PUn_8 | PUn_7 | PUn_6 | PUn_5 | PUn_4 | PUn_3 | PUn_2 | PUn_1 | PUn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.29 PUn/JPU0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PUn_[15:0] | Specifies whether a pull-up resistor is connected to the corresponding pin.<br>0: No pull-up resistor connected<br>1: Pull-up resistor connected |

**NOTES**

1. If a pin is configured such that both an internal pull-up resistor (PUn.PUn_m = 1) and pull-down resistor (PDn.PDn_m = 1) are connected, the pull-down resistor is automatically selected and the pull-up resistor is not connected.
2. The pull-up resistor has no effect when the pin is operated in output mode.
3. The control bits of the JTAG pull-up option register (JPU0) are JPU0_ [7:0].

## 2.5.16 PDn/JPD0 — Pull-down option Register

This register specifies whether to connect an internal pull-down resistor to an input pin.

**Access:** PDn: This register can be read/written in 16-bit units.
JPD0:This register can be read/written in 8-bit units.

**Address:** PDn: <PORTn_base> + $4010_H$ + n × $40_H$
JPD0: <JPORT0_base> + $4010_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PDn_15 | PDn_14 | PDn_13 | PDn_12 | PDn_11 | PDn_10 | PDn_9 | PDn_8 | PDn_7 | PDn_6 | PDn_5 | PDn_4 | PDn_3 | PDn_2 | PDn_1 | PDn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.30 PDn/JPD0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PDn_[15:0] | Specifies whether to connect an internal pull-down resistor to the corresponding pin:<br>0: No internal pull-down resistor connected<br>1: An internal pull-down resistor connected |

**NOTES**

1. If a pin is configured such that both an internal pull-up resistor (PUn.PUn_m = 1) and pull-down resistor (PDn.PDn_m = 1) are connected, the pull-down resistor is automatically selected and the pull-up resistor is not connected.
2. The internal pull-down resistor has no effect when the pin is operated in output mode.
3. The control bits of the JTAG pull-down option register (JPD0) are JPD0_[7:0].

## 2.5.17 PODCn/JPODC0 — Port Open Drain Control Register

This register selects push-pull or open-drain as output buffer function.

**Access:** This register can be read/written in 32-bit units.

**Address:** PODCn: <PORTn_base> + $4014_H$ + n × $40_H$
JPODC0: <JPORT0_base> + $4014_H$

**Value after reset:** 0000 $0000_H$*1

Note 1. The PODC0 register is 0000 $0400_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PODCn_15 | PODCn_14 | PODCn_13 | PODCn_12 | PODCn_11 | PODCn_10 | PODCn_9 | PODCn_8 | PODCn_7 | PODCn_6 | PODCn_5 | PODCn_4 | PODCn_3 | PODCn_2 | PODCn_1 | PODCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.31 PODCn/JPODC0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | PODCn_[15:0] | Specifies the output buffer function.<br>0: Push-pull<br>1: Open-drain |

## 2.5.18 PODCEn/JPODCE0 — Port Open Drain Control Expansion Register

This register selects the emulated P-channel Open Drain together with PODCn.

**Access:** This register can be read/written in 32-bit units.

**Address:** PODCEn: <PORTn_base> + $4038_H$ + n × $40_H$
JPODCE0: <JPORT0_base> + $4038_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PODCEn_15 | PODCEn_14 | PODCEn_13 | PODCEn_12 | PODCEn_11 | PODCEn_10 | PODCEn_9 | PODCEn_8 | PODCEn_7 | PODCEn_6 | PODCEn_5 | PODCEn_4 | PODCEn_3 | PODCEn_2 | PODCEn_1 | PODCEn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.32 PODCEn/JPODCE0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | PODCEn_[15:0] | See **Table 2.33, Port Open Drain Control Expansion** for the detailed operation on PODCE. |

**Table 2.33 Port Open Drain Control Expansion**

| PODCEn_m | PODCn_m | Function |
|---|---|---|
| 0 | 0 | push-pull |
| 0 | 1 | emulated N-channel Open Drain |
| 1 | 0 | push-pull |
| 1 | 1 | emulated P-channel Open Drain |

## 2.5.19 PDSCn/JPDSC0 — Port drive strength control register

This register specifies the output driver strength of the port pin. This function is also related to the fast mode (high drive strength) and slow mode (low drive strength) of the output buffer.

**Access:** This register can be read/written in 32-bit units.

**Address:** PDSCn: <PORTn_base> + $4018_H$ + n × $40_H$
JPDSC0: <JPORT0_base> + $4018_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PDSCn_15 | PDSCn_14 | PDSCn_13 | PDSCn_12 | PDSCn_11 | PDSCn_10 | PDSCn_9 | PDSCn_8 | PDSCn_7 | PDSCn_6 | PDSCn_5 | PDSCn_4 | PDSCn_3 | PDSCn_2 | PDSCn_1 | PDSCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.34 PDSCn/JPDSC0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | PDSCn_[15:0] | Specifies the port drive strength of the output buffer of the port pin.<br>0: Lower drive strength<br>1: High drive strength |

## 2.5.20 PUCCn/JPUCC0 — Port Universal Characteristic Control Register

This register expands the number of output buffer characteristics selection capability. If it is used with PDSC register, the maximum of 4 characteristics selection is possible.

**Access:** This register can be read/written in 32-bit units.

**Address:** PUCCn: <PORTn_base> + $4028_H$ + n × $40_H$
JPUCC0: <JPORT0_base> + $4028_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PUCCn_15 | PUCCn_14 | PUCCn_13 | PUCCn_12 | PUCCn_11 | PUCCn_10 | PUCCn_9 | PUCCn_8 | PUCCn_7 | PUCCn_6 | PUCCn_5 | PUCCn_4 | PUCCn_3 | PUCCn_2 | PUCCn_1 | PUCCn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.35 PUCCn/JPUCC0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | PUCCn_[15:0] | Register to increase the number of output buffer characteristics selection capability. |

**Table 2.36 Port Output buffer Characteristics Selection (Output Buffer Type GPIO)**

| PUCCn_m | PDSCn_m | Function |
|---|---|---|
| 0 | 0 | GPIO Drive Strength 4 |
| | 1 | GPIO Drive Strength 3 |
| 1 | 0 | GPIO Drive Strength 2 |
| | 1 | GPIO Drive Strength 1 |

**Table 2.37 Port Output buffer Characteristics Selection (Output Buffer Type HSIO)**

| PUCCn_m | PDSCn_m | Function |
|---|---|---|
| 0 | 0 | HSIO Drive Strength 4 |
| | 1 | HSIO Drive Strength 3 |
| 1 | 0 | HSIO Drive Strength 2 |
| | 1 | HSIO Drive Strength 1 |

## 2.5.21 PISn/JPIS0 — Port input buffer selection register

This register specifies the input buffer characteristics.

**Access:** PISn: This register can be read/written in 16-bit units.
JPIS0: This register can be read/written in 8-bit units.

**Address:** PISn: <PORTn_base> + $401C_H$ + n × $40_H$
JPIS0: <JPORT0_base> + $401C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PISn_15 | PISn_14 | PISn_13 | PISn_12 | PISn_11 | PISn_10 | PISn_9 | PISn_8 | PISn_7 | PISn_6 | PISn_5 | PISn_4 | PISn_3 | PISn_2 | PISn_1 | PISn_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.38 PISn/JPIS0 register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | PISn_[15:0] | Specifies the input buffer characteristic:<br>0: Type 1 (SHMT1)<br>1: Type 2 (SHMT4) |

## 2.5.22 PCRn_m/JPCR0_m — Port Control Register

By going through this register, it is possible to have access to the registers of each port group, and the individual pins is specified all functions by 1 PCR register setting.

**Access:** PCRn_m/JPCR0_m: This register can be read/written in 32-bit units

**Address:** PCRn_m: <PORTn_base>+2000H + n × $40_H$ + m × $4_H$
JPCR0_m: <JPORT0_base>+2000H + m × $4_H$

**Value after reset:** 0000 $0010_H$*1*2

Note 1. The PCR0_10 register is 1000 $0000_H$

Note 2. The JPCR0_4 register is 00000000H, because do not support JPM0_4 bit.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | PINV | — | PODC | PODCE | — | PUCC | PDSC | — | — | — | PIS | PU | PD | PBDC | PIBC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R/W | R/W | R | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | P | — | — | — | PPR | — | PMC | PIPC | PM | — | — | PFCE | PFC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R/W | R/W |

**Table 2.39 PCRn_m/JPCR0_m register contents (1/2)**

| Bit | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 30 | PINV | Same as the m bit of PINVn/JPINV0 register |
| 29 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 28 | PODC | Same as the m bit of PODCn/JPODC0 register |
| 27 | PODCE | Same as the m bit of PODCEn/JPODCE0 register |
| 26 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 25 | PUCC | Same as the m bit of PUCCn/JPUCC0 register |
| 24 | PDSC | Same as the m bit of PDSCn/JPDSC0 register |
| 23 to 21 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 20 | PIS | Same as the m bit of PISn/JPIS0 register |
| 19 | PU | Same as the m bit of PUn/JPU0 register |
| 18 | PD | Same as the m bit of PDn/JPD0 register |
| 17 | PBDC | Same as the m bit of PBDCn/JPBDC0 register |
| 16 | PIBC | Same as the m bit of PIBCn/JPIBC0 register |
| 15 to 13 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 12 | P | Same as the m bit of Pn/JP0 register |
| 11 to 9 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 8 | PPR | Same as the m bit of PPRn/JPPR0 register |
| 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

**Table 2.39 PCRn_m/JPCR0_m register contents (2/2)**

| Bit | Bit Name | Function |
|---|---|---|
| 6 | PMC | Same as the m bit of PMCn/JPMC0 register |
| 5 | PIPC | Same as the m bit of PIPCn register |
| 4 | PM | Same as the m bit of PMn/JPM0 register |
| 3, 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | PFCE | Same as the m bit of PFCEn/JPFCE0 register |
| 0 | PFC | Same as the m bit of PFCn register |

**Table 2.40 Relationship between Other Registers and PCR(Port Control Register)**

Setting all attributes per each pin

Setting up to 16 pins at the same time per each attribute

| | | PCR register | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PCRn_15 | PCRn_14 | PCRn_13 | PCRn_12 | ... | PCRn_3 | PCRn_2 | PCRn_1 | PCRn_0 | Bit |
| Other registers | – | – | – | – | – | – | – | – | – | – | 31 |
| | PINVn | PINVn_15 | PINVn_14 | PINVn_13 | PINVn_12 | ... | PINVn_3 | PINVn_2 | PINVn_1 | PINVn_0 | 30 |
| | – | – | – | – | – | ... | – | – | – | – | 29 |
| | PODCn | PODCn_15 | PODCn_14 | PODCn_13 | PODCn_12 | ... | PODCn_3 | PODCn_2 | PODCn_1 | PODCn_0 | 28 |
| | PODCEn | PODCEn_15 | PODCEn_14 | PODCEn_13 | PODCEn_12 | ... | PODCEn_3 | PODCEn_2 | PODCEn_1 | PODCEn_0 | 27 |
| | – | – | – | – | – | ... | – | – | – | – | 26 |
| | PUCCn | PUCCn_15 | PUCCn_14 | PUCCn_13 | PUCCn_12 | ... | PUCCn_3 | PUCCn_2 | PUCCn_1 | PUCCn_0 | 25 |
| | PDSCn | PDSCn_15 | PDSCn_14 | PDSCn_13 | PDSCn_12 | ... | PDSCn_3 | PDSCn_2 | PDSCn_1 | PDSCn_0 | 24 |
| | – | – | – | – | – | ... | – | – | – | – | 23 |
| | – | – | – | – | – | ... | – | – | – | – | 22 |
| | – | – | – | – | – | ... | – | – | – | – | 21 |
| | PISn | PISn_15 | PISn_14 | PISn_13 | PISn_12 | ... | PISn_3 | PISn_2 | PISn_1 | PISn_0 | 20 |
| | PUn | PUn_15 | PUn_14 | PUn_13 | PUn_12 | ... | PUn_3 | PUn_2 | PUn_1 | PUn_0 | 19 |
| | PDn | PDn_15 | PDn_14 | PDn_13 | PDn_12 | ... | PDn_3 | PDn_2 | PDn_1 | PDn_0 | 18 |
| | PBDCn | PBDCn_15 | PBDCn_14 | PBDCn_13 | PBDCn_12 | ... | PBDCn_3 | PBDCn_2 | PBDCn_1 | PBDCn_0 | 17 |
| | PIBCn | PIBCn_15 | PIBCn_14 | PIBCn_13 | PIBCn_12 | ... | PIBCn_3 | PIBCn_2 | PIBCn_1 | PIBCn_0 | 16 |
| | – | – | – | – | – | ... | – | – | – | – | 15 |
| | – | – | – | – | – | ... | – | – | – | – | 14 |
| | – | – | – | – | – | ... | – | – | – | – | 13 |
| | Pn | Pn_15 | Pn_14 | Pn_13 | Pn_12 | ... | Pn_3 | Pn_2 | Pn_1 | Pn_0 | 12 |
| | – | – | – | – | – | ... | – | – | – | – | 11 |
| | – | – | – | – | – | ... | – | – | – | – | 10 |
| | – | – | – | – | – | ... | – | – | – | – | 9 |
| | PPRn | PPRn_15 | PPRn_14 | PPRn_13 | PPRn_12 | ... | PPRn_3 | PPRn_2 | PPRn_1 | PPRn_0 | 8 |
| | | | | | | | | | | | 7 |
| | PMCn | PMCn_15 | PMCn_14 | PMCn_13 | PMCn_12 | ... | PMCn_3 | PMCn_2 | PMCn_1 | PMCn_0 | 6 |
| | PIPCn | PIPCn_15 | PIPCn_14 | PIPCn_13 | PIPCn_12 | ... | PIPCn_3 | PIPCn_2 | PIPCn_1 | PIPCn_0 | 5 |
| | PMn | PMn_15 | PMn_14 | PMn_13 | PMn_12 | ... | PMn_3 | PMn_2 | PMn_1 | PMn_0 | 4 |
| | – | – | – | – | – | ... | – | – | – | – | 3 |
| | – | – | – | – | – | ... | – | – | – | – | 2 |
| | PFCEn | PFCEn_15 | PFCEn_14 | PFCEn_13 | PFCEn_12 | ... | PFCEn_3 | PFCEn_2 | PFCEn_1 | PFCEn_0 | 1 |
| | PFCn | PFCn_15 | PFCn_14 | PFCn_13 | PFCn_12 | ... | PFCn_3 | PFCn_2 | PFCn_1 | PFCn_0 | 0 |

## 2.6 Port Setting Flow Example

Port setting flow examples are shown in this section.

The following figure indicates an example of setting a port group collectively.

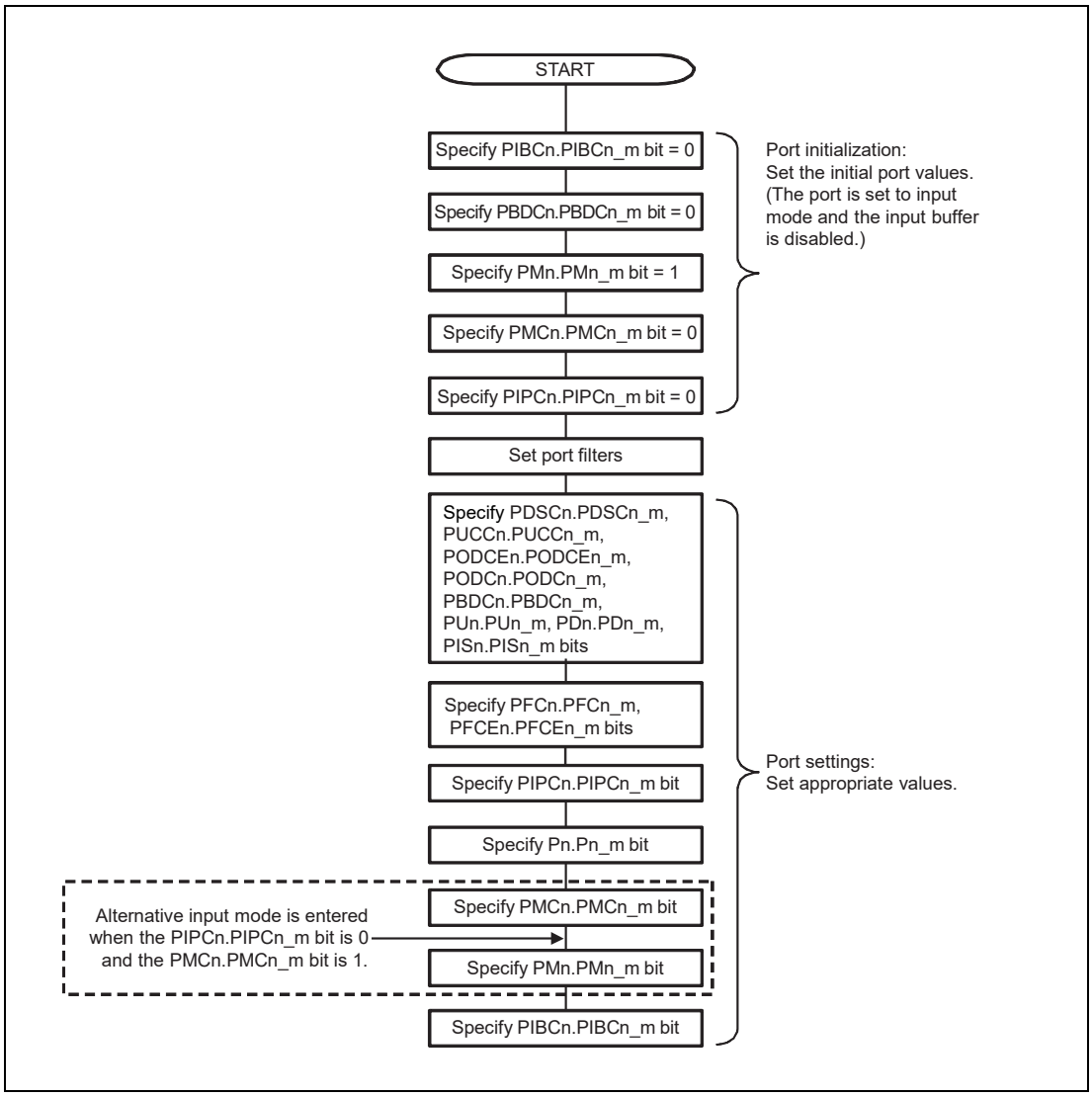



**Figure 2.9 Port setting flow example (setting collectively)**

### Setting individually

The following figure indicates an example of setting an individual port.

By going through this register, it is possible to have access to the registers of each port group, and the individual pins is specified all functions by 1 PCR register setting.

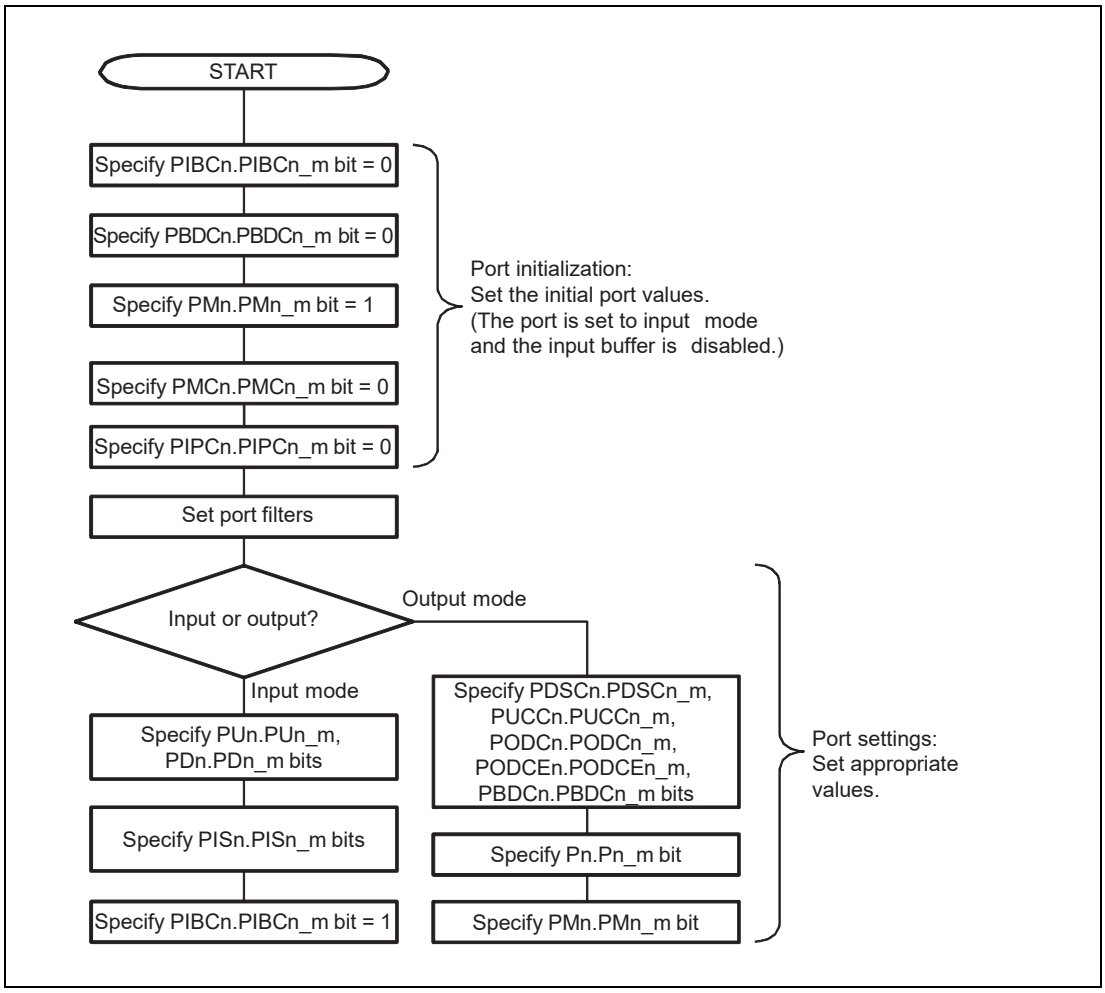



**Figure 2.10 Port setting flow example (Port mode)**

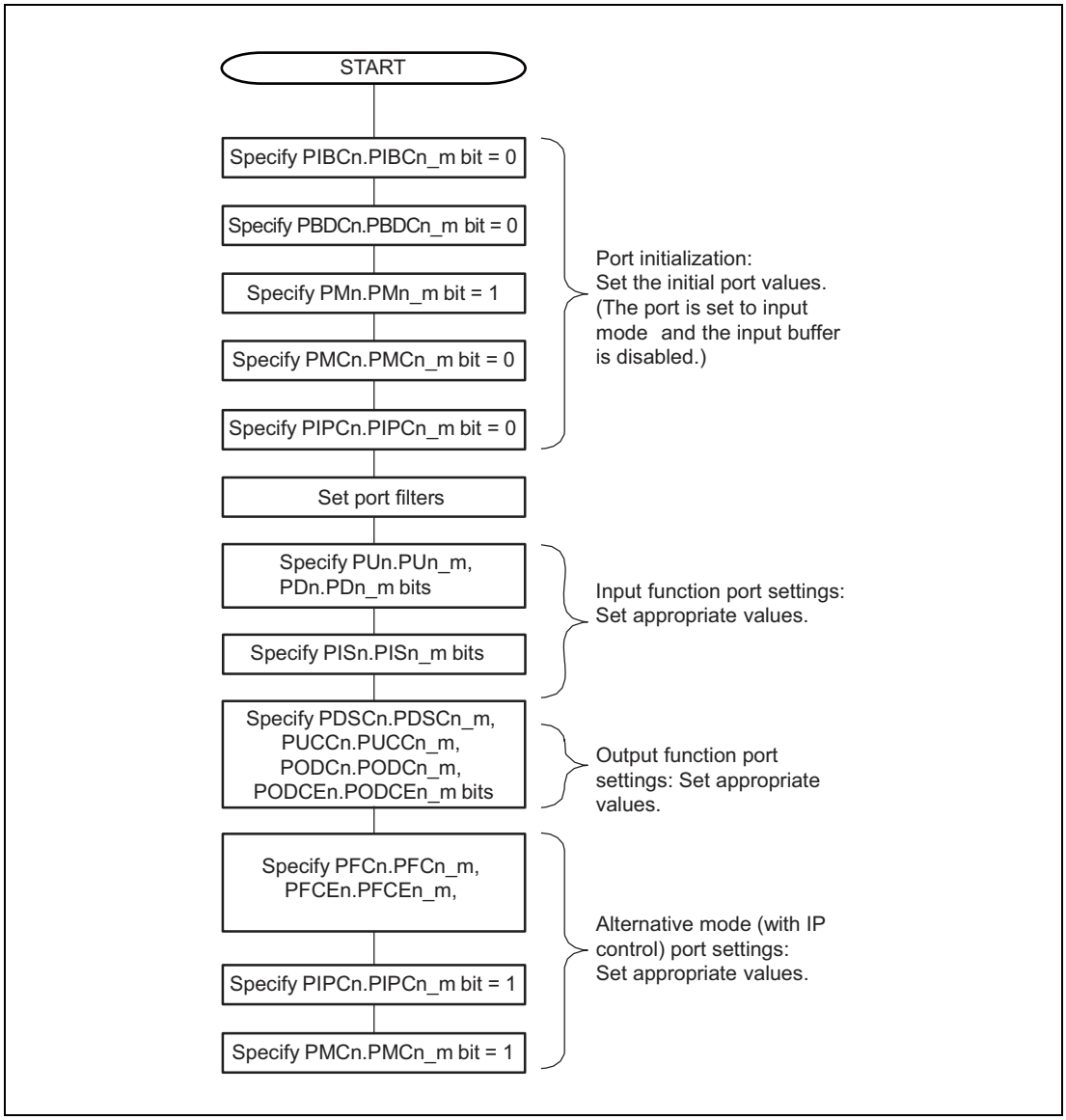



**Figure 2.11 Port setting flow example (Alternative mode w/ IP control)**

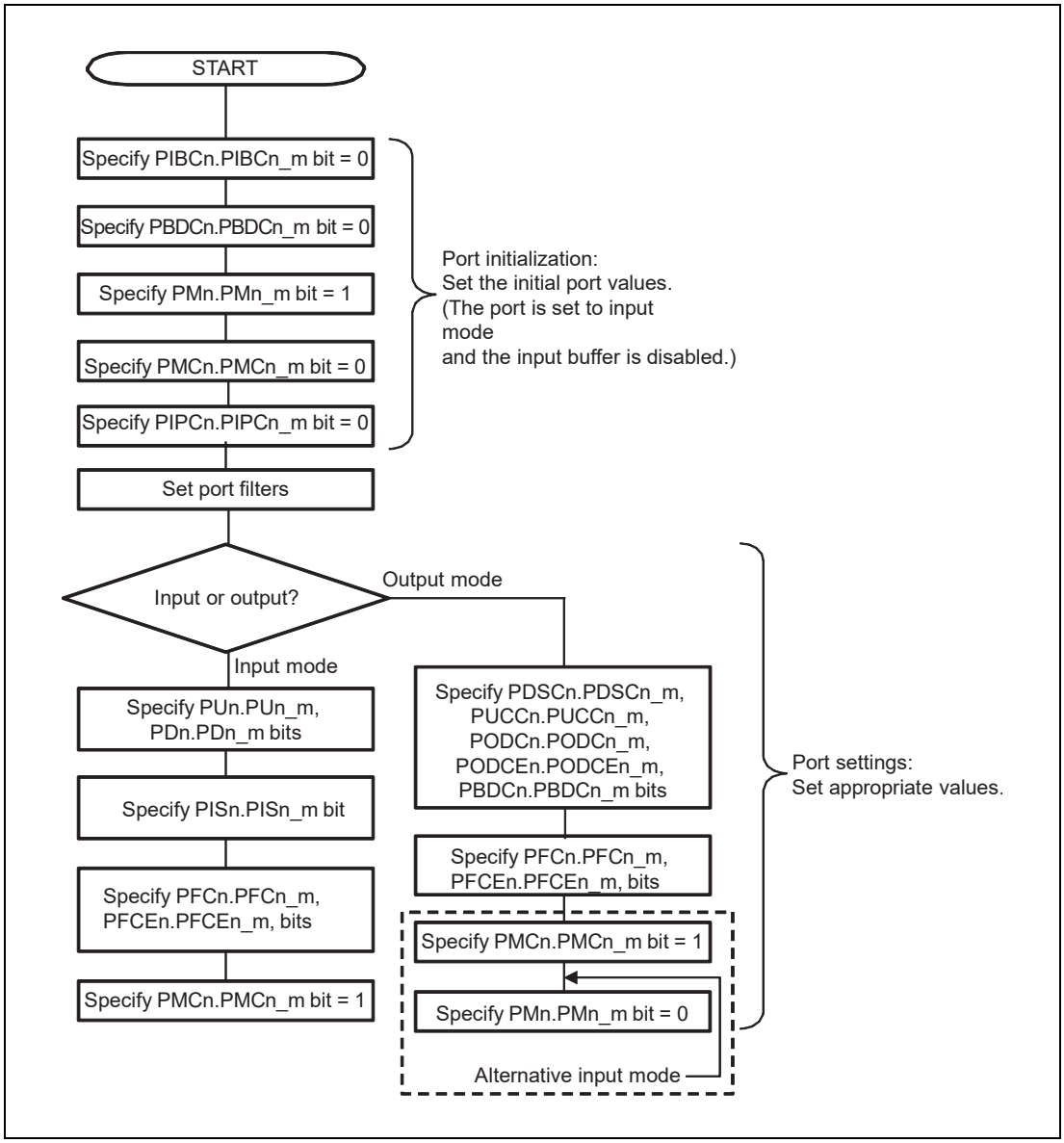



**Figure 2.12 Port setting flow example (Alternative mode w/o IP control)**

## 2.7 Noise Filter & Edge Detector

The input signals at some pins are passed through a filter to eliminate noise and glitches. This device supports both analog and digital filters. It also supports the function for edge and level detection after the signals have passed through a filter. The first part of this section provides an overview of port input signals that are equipped with a filter and the filter type, noise filter & edge detection control registers and bits, and register addresses.

Note: Level detection feature is not available in this device

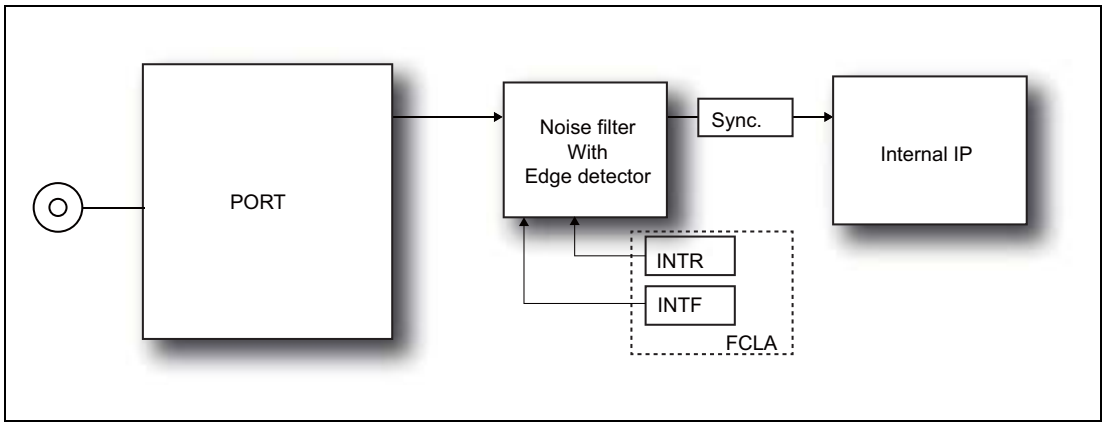



Figure 2.13 Noise Filter & Edge Detector Block Diagram

### 2.7.1 Port filter assignment

A list of the input pins that incorporate an analog or digital filter is provided below.

#### 2.7.1.1 Input pins that incorporate analog filter type A

The input pins of analog filter type A incorporate an analog filter and edge/level detection function. Edge/level detection is controlled by the following registers.

- Filter control register FCLAnCTLm (n = 0 to 6, m = 0 to 7)
  A dedicated FCLAnCTLm register is provided for each pin in a port that incorporates an analog filter.

Table 2.41 Input pins that incorporate analog filter type A

| | FCLAnCTLm Register Configuration | |
|---|---|---|
| **Input signal** | **Register** | **Address** |
| NMI | FCLAnCTLm | See **Table 2.44, Noise Filter Register List** |
| INTPn | FCLAnCTLm | See **Table 2.44, Noise Filter Register List** |

Here n is the external interrupt number, m varies from 0 to 7 and x varies from L & H

#### 2.7.1.2 Input pins that incorporate analog filter type B

The input pins of analog filter type B only incorporate the analog filter function.

Table 2.42 Input pins that incorporate analog filter type B

| Input Signal |
|---|
| FLMD0 |
| FLMD1 |
| MODE0 |
| MODE1 |
| RESETZ |
| TRSTZ |
| AURORES1Z |
| AURORESPDZ |
| AURORES2Z |
| ERAMRESPDZ |
| ERAMRES2Z |

#### 2.7.1.3 Input pins that incorporate digital filter type C

The input pins of digital filter type C incorporate a digital filter and edge detection function. The digital filter and edge detection are controlled by the following registers.

- Filter control register FCLAnCTLm (m = 0)
  Each port with a digital filter has a special FCLAnCTLm register.
- Digital noise elimination control register DNFAnCTL
  Each DNFAnCTL control register controls digital filter processing for three input signals per group.
- Digital noise elimination enable register DNFAnEN
  The setting of the DNFAnNFENL[2:0] bits in DNFAnEN enables or disables digital noise elimination for three input signals per group.

Input pins that incorporate digital filter type C are yet to be decided.

#### 2.7.1.4 Input pins that incorporate digital filter type D

The input pins of digital filter type D incorporate a digital filter and without edge detection function. The digital filter is controlled by registers same as type C.

#### 2.7.1.5 Input pins that incorporate combination of analog filter and digital filter type E

The input pins of filter type E incorporate a digital filter and analog filter without edge detection function. The analog filter and digital filter is controlled by registers same as type A and C.

## 2.7.2 Noise Filter Assignment

The noise filter type of each pin is shown below.

**Table 2.43 Noise Filter Assignment Overview (1/3)**

| Function | | Port | | | Noise Filter | | | | | Detector Type | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Digital noise elimination enable register | | | | | |
| Group | Name | Name | ALT_No | Active Level | Filter type | Filter Control Register | Digital noise elimination control register | register | bit | Rise | Fall | Both | Level |
| System Control | RESETZ | — | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |
| | TRSTZ | JP0_4 | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |
| | FLMD0 | — | — | H | ANF (typeB) | — | — | — | — | — | — | — | — |
| | FLMD1 | P4_5 | — | H | ANF (typeB) | — | — | — | — | — | — | — | — |
| | MODE0 | P4_2 | — | H | ANF (typeB) | — | — | — | — | — | — | — | — |
| | MODE1 | P4_3 | — | H | ANF (typeB) | — | — | — | — | — | — | — | — |
| | AURORES1Z | — | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |
| | AURORESPDZ | — | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |
| | AURORES2Z | — | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |
| | ERAMRESPDZ | — | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |
| | ERAMRESZ2 | — | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |
| Interrupt | NMI | P5_7 | ALT_3 | H | ANF (typeA) | FCLA0CTL0 | — | — | — | √ | √ | √ | √ |
| | INTP0 | P4_7 | ALT_3 | H | ANF (typeA) | FCLA0CTL1 | — | — | — | √ | √ | √ | √ |
| | INTP1 | P4_12 | ALT_2 | H | ANF (typeA) | FCLA0CTL2 | — | — | — | √ | √ | √ | √ |
| | INTP2 | P5_13 | ALT_3 | H | ANF (typeA) | FCLA0CTL3 | — | — | — | √ | √ | √ | √ |
| | INTP3 | P0_5 | ALT_2 | H | ANF (typeA) | FCLA0CTL4 | — | — | — | √ | √ | √ | √ |
| | INTP4 | P1_1 | ALT_2 | H | ANF (typeA) | FCLA0CTL5 | — | — | — | √ | √ | √ | √ |
| | INPT5 | P2_7 | ALT_2 | H | ANF (typeA) | FCLA0CTL6 | — | — | — | √ | √ | √ | √ |
| | INTP6 | P3_5 | ALT_3 | H | ANF (typeA) | FCLA0CTL7 | — | — | — | √ | √ | √ | √ |
| | INTP7 | P3_8 | ALT_3 | H | ANF (typeA) | FCLA1CTL0 | — | — | — | √ | √ | √ | √ |
| | INTP8 | P8_0 | ALT_3 | H | ANF (typeA) | FCLA1CTL1 | — | — | — | √ | √ | √ | √ |
| | INTP9 | P6_4 | ALT_4 | H | ANF (typeA) | FCLA1CTL2 | — | — | — | √ | √ | √ | √ |
| | INTP10 | P3_14 | ALT_4 | H | ANF (typeA) | FCLA1CTL3 | — | — | — | √ | √ | √ | √ |
| | INTP11 | P7_7 | ALT_3 | H | ANF (typeA) | FCLA1CTL4 | — | — | — | √ | √ | √ | √ |

Table 2.43 Noise Filter Assignment Overview (2/3)

| Function | | Port | | | Noise Filter | | | | | Detector Type | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Group | Name | Name | ALT_No | Active Level | Filter type | Filter Control Register | Digital noise elimination control register | Digital noise elimination enable register | | Rise | Fall | Both | Level |
| | | | | | | | | register | bit | | | | |
| Wake Up (RLIN)*1 | RLIN30RX | P2_9 | ALT_3 | H | DNF (typeC) | FCLA2CTL0 | DNFA2CTL | DNFA2EN (DNFA2ENL) | bit0 | √ | √ | √ | — |
| | | P5_15 | ALT_3 | H | | | | | | | | | |
| | | P4_8 | ALT_3 | H | | | | | | | | | |
| | | P5_9 | ALT_2 | H | | | | | | | | | |
| | | P2_6 | ALT_2 | H | | | | | | | | | |
| | | P3_14 | ALT_2 | H | | | | | | | | | |
| | | P3_2 | ALT_2 | H | | | | | | | | | |
| | | P5_13 | ALT_4 | H | | | | | | | | | |
| | | P4_5 | ALT_4 | H | | | | | | | | | |
| | RLIN31RX | P2_4 | ALT_3 | H | DNF (typeC) | FCLA2CTL1 | DNFA2CTL | DNFA2EN (DNFA2ENL) | bit1 | √ | √ | √ | — |
| | | P5_4 | ALT_4 | H | | | | | | | | | |
| Wake Up (MTTCAN) *1 | MTTCAN0RX | P2_6 | ALT_3 | H | DNF (typeC) | FCLA3CTL0 | DNFA3CTL | DNFA3EN (DNFA3ENL) | bit0 | √ | √ | √ | — |
| | | P4_5 | ALT_3 | H | | | | | | | | | |
| | | P5_15 | ALT_2 | H | | | | | | | | | |
| Wake Up (MCAN)*1 | MCAN0RX | P0_13 | ALT_4 | H | DNF (typeC) | FCLA3CTL1 | DNFA3CTL | DNFA3EN (DNFA3ENL) | bit1 | √ | √ | √ | — |
| | | P3_14 | ALT_3 | H | | | | | | | | | |
| | | P5_0 | ALT_3 | H | | | | | | | | | |
| Wake Up (FLX)*1 | FLX0RXDA | P3_2 | ALT_4 | H | DNF (typeC) | FCLA4CTL0 | DNFA4CTL | DNFA4EN (DNFA4ENL) | bit0 | √ | √ | √ | — |
| | | P3_14 | ALT_4 | H | | | | | | | | | |
| | FLX1RXDA | P7_7 | ALT_2 | H | DNF (typeC) | FCLA4CTL1 | DNFA4CTL | DNFA4EN (DNFA4ENL) | bit1 | √ | √ | √ | — |
| SENT | SENT0RX | P0_0 | ALT_3 | H | ANF/DNF (TypeE) | FCLA5CTL0 | DNFA5CTL | DNFA5EN (DNFA5ENL) | bit0 | — | — | — | — |
| | | P0_14 | ALT_3 | H | | | | | | | | | |
| | | P3_2 | ALT_3 | H | | | | | | | | | |
| | | P4_7 | ALT_4 | H | | | | | | | | | |
| | SENT1RX | P0_1 | ALT_3 | H | ANF/DNF (TypeE) | FCLA5CTL1 | DNFA5CTL | DNFA5EN (DNFA5ENL) | bit1 | — | — | — | — |
| | | P3_3 | ALT_3 | H | | | | | | | | | |
| | | P4_8 | ALT_4 | H | | | | | | | | | |
| | SENT2RX | P3_12 | ALT_4 | H | ANF/DNF (TypeE) | FCLA5CTL2 | DNFA5CTL | DNFA5EN (DNFA5ENL) | bit2 | — | — | — | — |
| | | P4_13 | ALT_3 | H | | | | | | | | | |
| | | P5_13 | ALT_2 | H | | | | | | | | | |
| | SENT3RX | P3_13 | ALT_4 | H | ANF/DNF (TypeE) | FCLA5CTL3 | DNFA5CTL | DNFA5EN (DNFA5ENL) | bit3 | — | — | — | — |
| | | P4_14 | ALT_3 | H | | | | | | | | | |
| | | P5_12 | ALT_4 | H | | | | | | | | | |
| | SENT4RX | P0_13 | ALT_3 | H | ANF/DNF (TypeE) | FCLA5CTL4 | DNFA5CTL | DNFA5EN (DNFA5ENL) | bit4 | — | — | — | — |
| | | P6_0 | ALT_3 | H | | | | | | | | | |
| | | P6_2 | ALT_3 | H | | | | | | | | | |
| | SENT5RX | P6_1 | ALT_3 | H | ANF/DNF (TypeE) | FCLA5CTL5 | DNFA5CTL | DNFA5EN (DNFA5ENL) | bit5 | — | — | — | — |
| | | P6_3 | ALT_3 | H | | | | | | | | | |
| | SENT6RX | P6_4 | ALT_3 | H | ANF/DNF (TypeE) | FCLA5CTL6 | DNFA5CTL | DNFA5EN (DNFA5ENL) | bit6 | — | — | — | — |
| | | P7_1 | ALT_3 | H | | | | | | | | | |
| | SENT7RX | P6_5 | ALT_3 | H | ANF/DNF (TypeE) | FCLA5CTL7 | DNFA5CTL | DNFA5EN (DNFA5ENL) | bit7 | — | — | — | — |
| | | P7_0 | ALT_3 | H | | | | | | | | | |
| | SENT8RX | P9_1 | ALT_4 | H | ANF/DNF (TypeE) | FCLA6CTL0 | DNFA6CTL | DNFA6EN (DNFA6ENL) | bit0 | — | — | — | — |
| | SENT9RX | P9_3 | ALT_4 | H | ANF/DNF (TypeE) | FCLA6CTL1 | DNFA6CTL | DNFA6EN (DNFA6ENL) | bit1 | — | — | — | — |

**Table 2.43 Noise Filter Assignment Overview (3/3)**

| Function | | Port | | | Noise Filter | | | | | Detector Type | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Group | Name | Name | ALT_No | Active Level | Filter type | Filter Control Register | Digital noise elimination control register | Digital noise elimination enable register | | Rise | Fall | Both | Level |
| | | | | | | | | register | bit | | | | |
| ADCA | ADC0TRG | P5_4 | ALT_3 | H | DNF (typeD) | — | DNFA7CTL | DNFA7EN (DNFA7ENL) | bit0 | — | — | — | — |
| | ADC1TRG | P4_12 | ALT_4 | H | DNF (typeD) | — | DNFA7CTL | DNFA7EN (DNFA7ENL) | bit1 | — | — | — | — |
| Other | AURORES1Z | — | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |
| | AURORESPDZ | — | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |
| | AURORES2Z | — | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |
| | ERAMRESPDZ | — | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |
| | ERAMRESZ2 | — | — | L | ANF (typeB) | — | — | — | — | — | — | — | — |

Note 1. RXD doesn't have noise filter. DNF and ED is inserted only path for wake up factor.

Note 2. GTM has DNF for each input within itself. For more details, see **Section 25.10, Timer Input Module (TIM)**.

About edge detect of each serial macro's received data for Wake up, additional circuit diagram is described as following. DNFAm/FICTLm block of each Figure is added for the edge detect function to wake up.

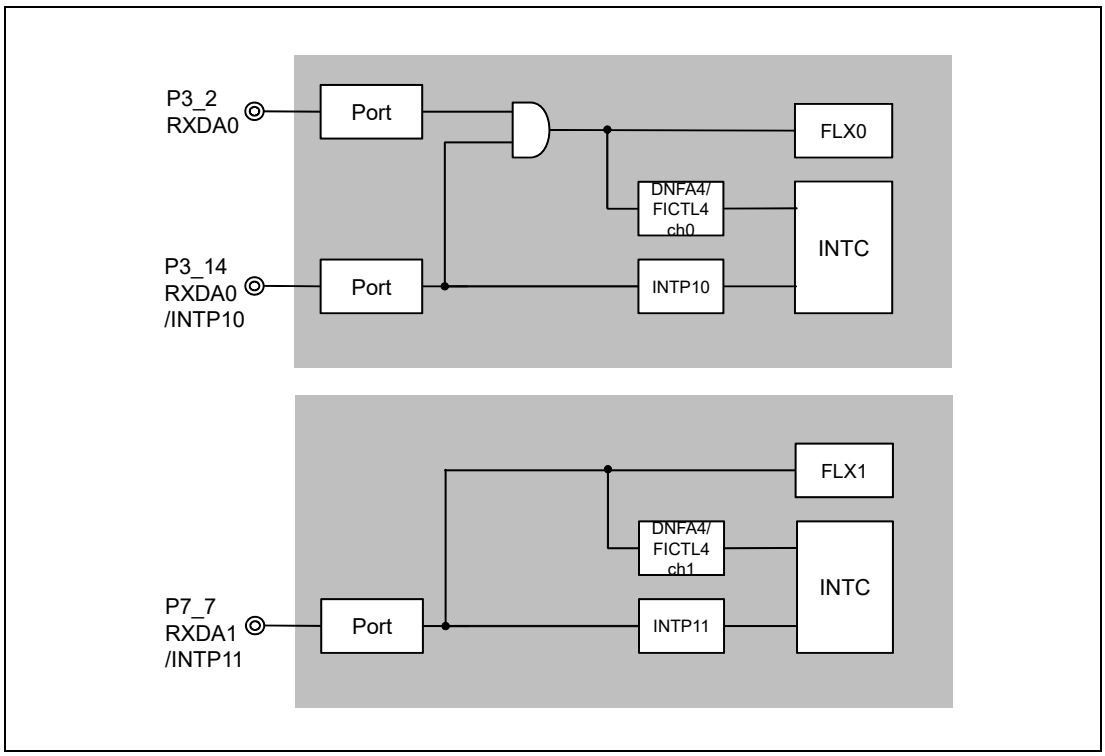



**Figure 2.14 FlexRay RXDA and INTP connection diagram**

**NOTE**

When RXDA0 of P3_2 is used for FLX0, P3_14 can not be used as INTP10 for Wake up.

When RXDA0 of P3_14 is used for FLX0, P3_14 can be used as INTP10 for Wake up.

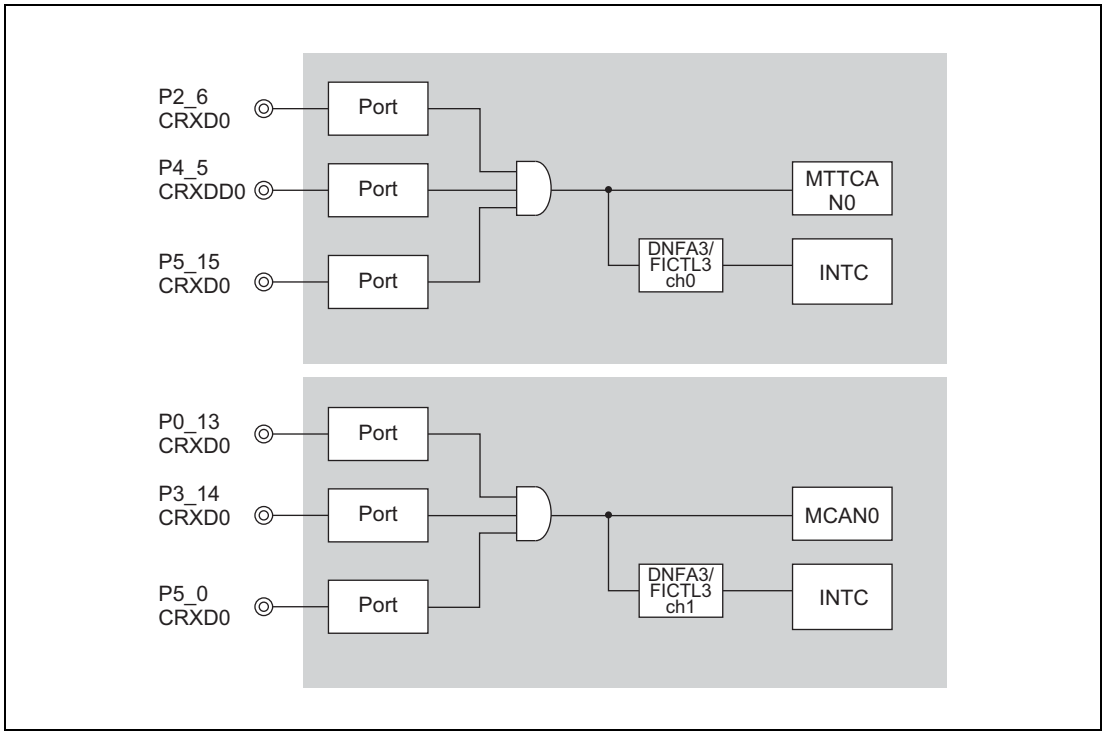



**Figure 2.15 MTTCAN0/MCAN0 CRXD connection diagram**

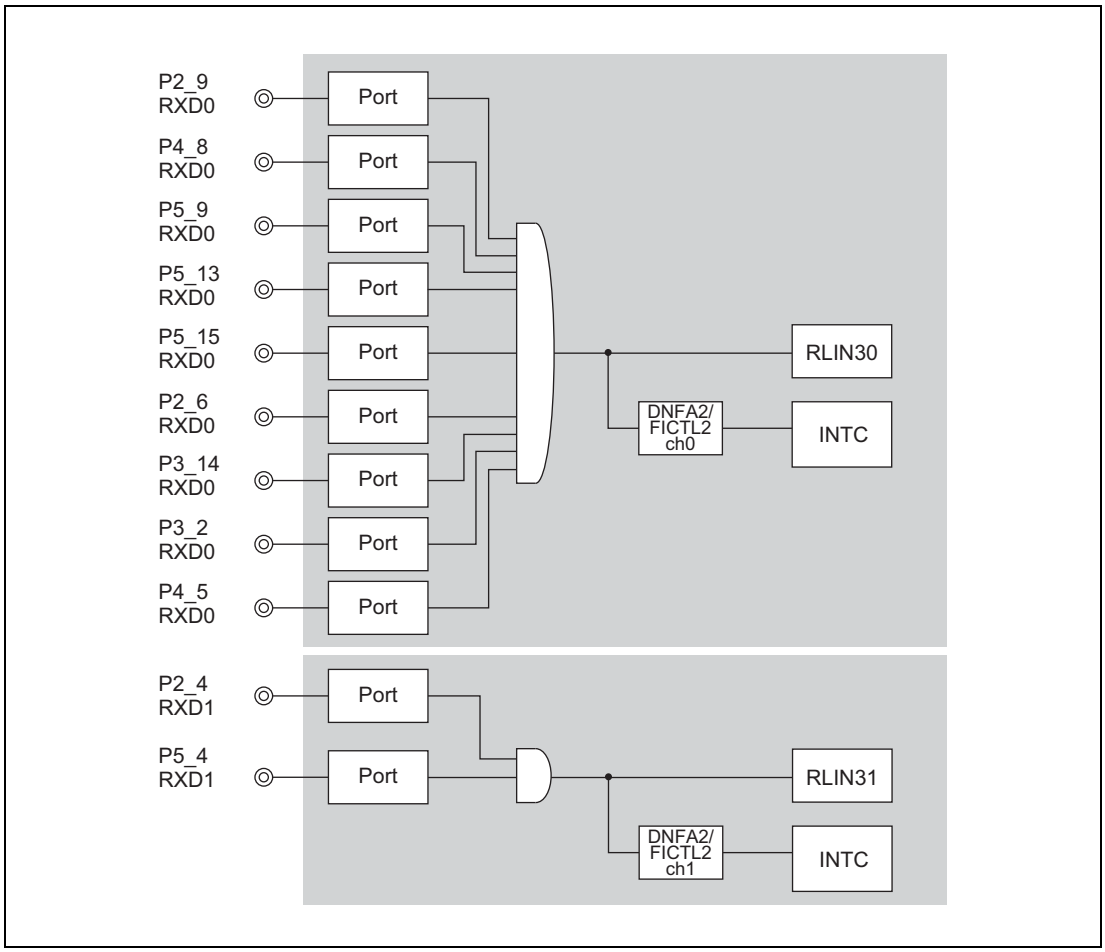



**Figure 2.16 RLIN3 RXD connection diagram**

### 2.7.2.1 Noise Filter Register

Table 2.44 Noise Filter Register List

| Register Name | Function | R/W | Reset Value | Access Unit | | | Base Address | Offset Address |
|---|---|---|---|---|---|---|---|---|
| | | | | 8 bit | 16 bit | 32 bit | | |
| DNFA2CTL | Digital noise elimination control register | R/W | 00H | √ | — | — | FFC30200H | 00H |
| DNFA2EN | Digital noise elimination enable register | R/W | 0000H | — | √ | — | | 04H |
| DNFA2ENL | Digital noise elimination enable L register | R/W | 00H | √ | — | — | | 0CH |
| DNFA3CTL | Digital noise elimination control register | R/W | 00H | √ | — | — | FFC30300H | 00H |
| DNFA3EN | Digital noise elimination enable register | R/W | 0000H | — | √ | — | | 04H |
| DNFA3ENL | Digital noise elimination enable L register | R/W | 00H | √ | — | — | | 0CH |
| DNFA4CTL | Digital noise elimination control register | R/W | 00H | √ | — | — | FFC30400H | 00H |
| DNFA4EN | Digital noise elimination enable register | R/W | 0000H | — | √ | — | | 04H |
| DNFA4ENL | Digital noise elimination enable L register | R/W | 00H | √ | — | — | | 0CH |
| DNFA5CTL | Digital noise elimination control register | R/W | 00H | √ | — | — | FFC30500H | 00H |
| DNFA5EN | Digital noise elimination enable register | R/W | 0000H | — | √ | — | | 04H |
| DNFA5ENL | Digital noise elimination enable L register | R/W | 00H | √ | — | — | | 0CH |
| DNFA6CTL | Digital noise elimination control register | R/W | 00H | √ | — | — | FFC30600H | 00H |
| DNFA6EN | Digital noise elimination enable register | R/W | 0000H | — | √ | — | | 04H |
| DNFA6ENL | Digital noise elimination enable L register | R/W | 00H | √ | — | — | | 0CH |
| DNFA7CTL | Digital noise elimination control register | R/W | 00H | √ | — | — | FFC30700H | 00H |
| DNFA7EN | Digital noise elimination enable register | R/W | 0000H | — | √ | — | | 04H |
| DNFA7ENL | Digital noise elimination enable L register | R/W | 00H | √ | — | — | | 0CH |
| FCLA0CTL0 | Filter Control register 0 | R/W | 00H | √ | — | — | FFC34000H | 00H |
| FCLA0CTL1 | Filter Control register 1 | R/W | 00H | √ | — | — | | 04H |
| FCLA0CTL2 | Filter Control register 2 | R/W | 00H | √ | — | — | | 08H |
| FCLA0CTL3 | Filter Control register 3 | R/W | 00H | √ | — | — | | 0CH |
| FCLA0CTL4 | Filter Control register 4 | R/W | 00H | √ | — | — | | 10H |
| FCLA0CTL5 | Filter Control register 5 | R/W | 00H | √ | — | — | | 14H |
| FCLA0CTL6 | Filter Control register 6 | R/W | 00H | √ | — | — | | 18H |
| FCLA0CTL7 | Filter Control register 7 | R/W | 00H | √ | — | — | | 1CH |
| FCLA1CTL0 | Filter Control register 0 | R/W | 00H | √ | — | — | FFC34100H | 00H |
| FCLA1CTL1 | Filter Control register 1 | R/W | 00H | √ | — | — | | 04H |
| FCLA1CTL2 | Filter Control register 2 | R/W | 00H | √ | — | — | | 08H |
| FCLA1CTL3 | Filter Control register 3 | R/W | 00H | √ | — | — | | 0CH |
| FCLA1CTL4 | Filter Control register 4 | R/W | 00H | √ | — | — | | 10H |
| FCLA2CTL0 | Filter Control register 0 | R/W | 00H | √ | — | — | FFC34200H | 00H |
| FCLA2CTL1 | Filter Control register 1 | R/W | 00H | √ | — | — | | 04H |
| FCLA3CTL0 | Filter Control register 0 | R/W | 00H | √ | — | — | FFC34300H | 00H |
| FCLA3CTL1 | Filter Control register 1 | R/W | 00H | √ | — | — | | 04H |
| FCLA4CTL0 | Filter Control register 0 | R/W | 00H | √ | — | — | FFC34400H | 00H |
| FCLA4CTL1 | Filter Control register 1 | R/W | 00H | √ | — | — | | 04H |
| FCLA5CTL0 | Filter Control register 0 | R/W | 00H | √ | — | — | FFC34500H | 00H |
| FCLA5CTL1 | Filter Control register 1 | R/W | 00H | √ | — | — | | 04H |
| FCLA5CTL2 | Filter Control register 2 | R/W | 00H | √ | — | — | | 08H |
| FCLA5CTL3 | Filter Control register 3 | R/W | 00H | √ | — | — | | 0CH |
| FCLA5CTL4 | Filter Control register 4 | R/W | 00H | √ | — | — | | 10H |
| FCLA5CTL5 | Filter Control register 5 | R/W | 00H | √ | — | — | | 14H |
| FCLA5CTL6 | Filter Control register 6 | R/W | 00H | √ | — | — | | 18H |
| FCLA5CTL7 | Filter Control register 7 | R/W | 00H | √ | — | — | | 1CH |
| FCLA6CTL0 | Filter Control register 0 | R/W | 00H | √ | — | — | FFC34600H | 00H |
| FCLA6CTL1 | Filter Control register 1 | R/W | 00H | √ | — | — | | 04H |

### 2.7.2.2 DNFAnEN Register List

Table 2.45 DNFAnEN Register List

| Register Name | bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DNFA2EN | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RLIN31RX | RLIN30RX |
| DNFA3EN | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MCAN0RX | MTTCAN0RX |
| DNFA4EN | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FLX1RXDA | FLX0RXDA |
| DNFA5EN | — | — | — | — | — | — | — | — | SENT7RX | SENT6RX | SENT5RX | SENT4RX | SENT3RX | SENT2RX | SENT1RX | SENT0RX |
| DNFA6EN | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SENT9RX | SENT8RX |
| DNFA7EN | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ADTRG1 | ADTRG0 |

## 2.7.3 Description of Port Noise Filter & Edge/Level Detection

External input signals pass through different types of filters according to the application of the signal to be filtered.

### (1) Analog filters

Analog filters have fixed characteristics.

- Type A: An analog filter with edge detection or level detection.
  Used for external interrupt signals.
- Type B: An analog filter only.
  Used for the external RESETZ input and mode signals.

### (2) Digital filters

The digital filter characteristics can be adjusted to suit the application.

- Type C: A digital filter with edge detection.
  Used for the wake up signal.
- Type D: A digital filter without edge detection.
  Used for the ADC trigger input signal.

### (3) Digital and Analog filters

- Type E: An analog and digital filter with edge detection.
  Used for the SENT receive signal.

### 2.7.3.1 Analog filters

#### Analog filter characteristic

The characteristics of the analog filter as well as of the level and edge detectors are specified in the **Section 35, Electrical Specifications**.

#### Analog filters control registers

For each input signal, that is equipped with an analog filter, a dedicated control register FCLAnCTLm and a control register in each peripheral macro are provided.

### (1) Analog filter type A

A block diagram of analog filter type A is shown below.

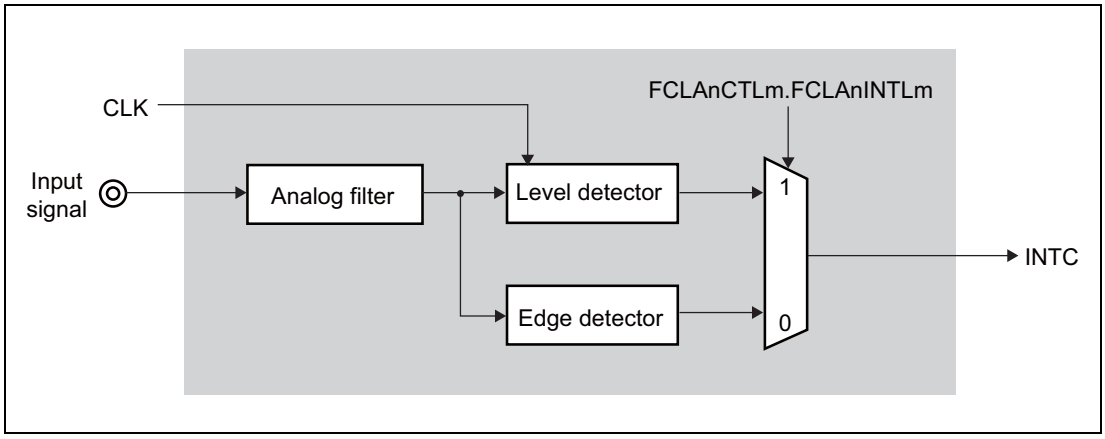



**Figure 2.17 Block diagram of analog filter type A**

After passing an external signal through the filter to eliminate noise and spikes, the filter generates an output signal according to whether an event is detected; that is whether a specified level is detected or whether a change in the level (an edge) occurs.

Whether a level or an edge is detected is selected by the control bit FCLAnCTLm.FCLAnINTLm.

- FCLAnCTLm = 0: Edge detection
  Whether a rising or falling edge is detected can be specified by setting the FCLAnCTLm.FCLAnINTRm and FCLAnCTLm.FCLAnINTFm bits.
- FCLAnINTLm = 1: Level detection
  The detection of a high level or low level can be specified by setting FCLAnCTLm.FCLAnINTRm.

The table below summarizes the detection conditions of the analog filter.

**Table 2.46 Analog filter event detection conditions**

| FCLAnINTLm | FCLAnINTFm | FCLAnINTRm | Edge detection | Level detection |
|---|---|---|---|---|
| 0 | 0 | 0 | No edge detected | Disabled |
| | 0 | 1 | Rising edge | |
| | 1 | 0 | Falling edge | |
| | 1 | 1 | Rising and falling edges | |
| 1 | X | 0 | Disabled | Low Level |
| | X | 1 | | High Level |

**(2) Analog filter type B**

A block diagram of analog filter type B is shown below.

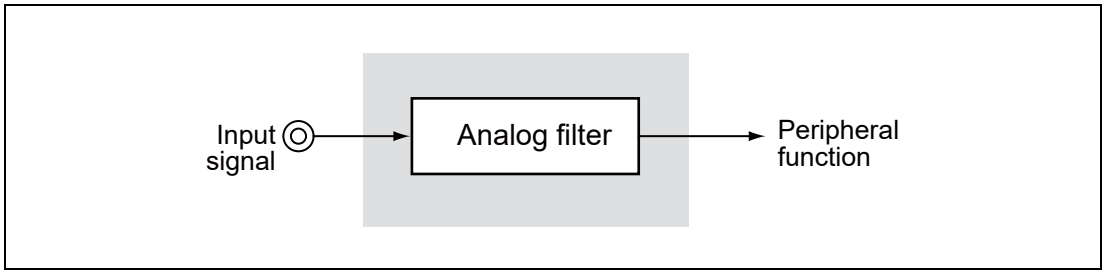



**Figure 2.18 Block diagram of analog filter type B**

The occurred signals are always signals that have passed through an analog filter.

### 2.7.3.2 Digital filters

#### Digital filter characteristic

The digital filters allow adjusting the filter characteristics to the needs of the application.

The input signal is sampled with the sampling frequency $f_s$.

If a specified number of successive samples yield the same (high or low) level, the signal level is judged as valid and the filter output signal is set accordingly.

If an external signal level change is detected within the specified number of samples (same level samples s), the signal level is judged as noise – or a spike – and the filter output signal does not change.

The length of an external signal pulse to be judged as noise depends on the sampling frequency and the specified number of same level samples.

Both parameters can be specified:

- DNFAnCTL.DNFAnPRS[2:0] allows to select the sampling frequency to
  $f_s = f_{DNFATCKI} / 2^{DNFAnPRS[2:0]}$
  where $f_{DNFATCKI}$ is the frequency of the DNFATCKI clock.
- DNFAnCTL.DNFAnNFSTS[1:0] determines the number s of same level samples (2 to 5):
  External signal pulses, shorter than
  DNFAnNFSTS[1:0] × 1/ $f_s$
  are always suppressed.
  External signal pulses, longer than
  (DNFAnNFSTS[1:0] + 1) × 1/ $f_s$
  are always judged as valid and are passed on to the filter output.
  Consequently, external signal pulses in the range
  DNFAnNFSTS[1:0] × 1/ $f_s$ to (DNFAnNFSTS[1:0] + 1) × 1/ $f_s$
  may be suppressed or judged as valid.
  The filter operation is illustrated in the figure below with DNFAnNFSTS[1:0] = $01_B$, i.e. s = 3 same level samples.

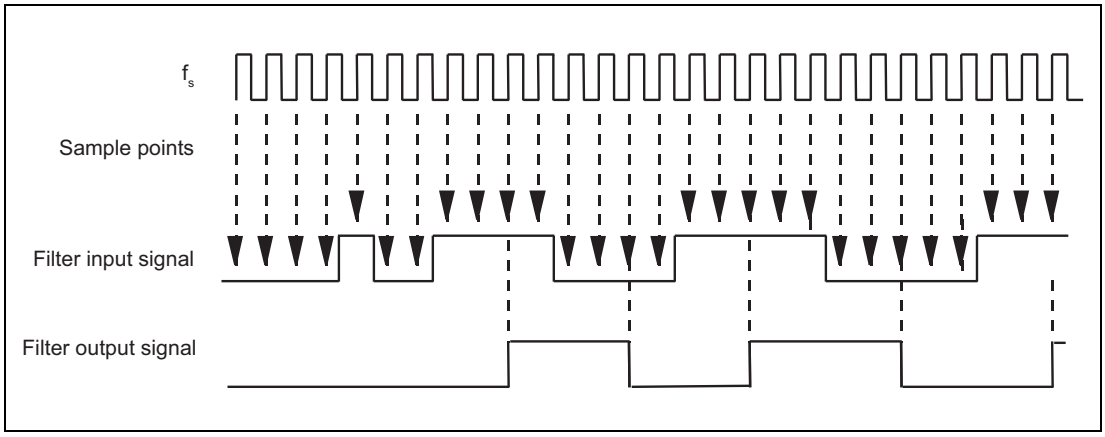



**Figure 2.19 Digital Filter function**

### Digital filter groups

The input signals with digital filters are ordered in groups of up to 16 signals.

The digital filter characteristics, specified by DNFAnCTL.DNFAnPRS[2:0] and DNFAnNFSTS[1:0] apply to the filters of the entire group.

However the digital filter for each signal can be enabled or disabled separately by DNFAnEN.DNFAnNFENm.

**CAUTIONS**

1. **When the output signal from the digital filter is set to an input for an alternative function, allow at least the following interval to elapse after the digital filter is enabled (DNFAnEN.DNFAnENm = 1) for the port pin to switch to the alternative function.**
   **DNFAnNFSTS[1:0] × $1/f_s$ + 4 × $1/f_{DNFATCKI}$**
2. **When a digital filter is used with an interrupt acting as an event output signal, only enable the digital filter (by setting DNFAnEN.DNFAnENm = 1) while interrupts are disabled. Furthermore, only enable interrupts after enabling the digital filter, waiting for the time below to elapse, and then clearing the interrupt request flag.**
   **DNFAnNFSTS[1:0] × $1/f_s$ + 5 × $1/f_{DNFATCKI}$**

### Digital filters control registers

For each group of up to 16 digital filters a common digital filter setup register DNFAnCTL and digital filter enable register DNFAnEN are provided with the same index n.

While the filter setup by DNFAnCTL affects the entire group, the control bits DNFAnENm in the filter enable register DNFAnEN allows to enable or disable each filter separately. The register index m is in the range from 0 to 15:

DNFAnCTL is the control register of group n for the digital filters m (m = 0 to 15), enabled/disabled by the DNFAnEN.DNAFnEN0 to DNFAnEN.DNAFnEN15 control bits.

The edge detection setup is done via the filter dedicated control register and the registers for individual peripheral functions.

The assignment of the input signals to the control registers and their addresses are given in **Table 2.44, Noise Filter Register List**.

**CAUTION**

**Do not change any control register settings, while the concerned digital filter is enabled by DNFAnEN.DNAFnENm = 1. Otherwise an unintended filter output may be generated.**

### (1) Digital filter type C

A block diagram of digital filter type C is shown below.

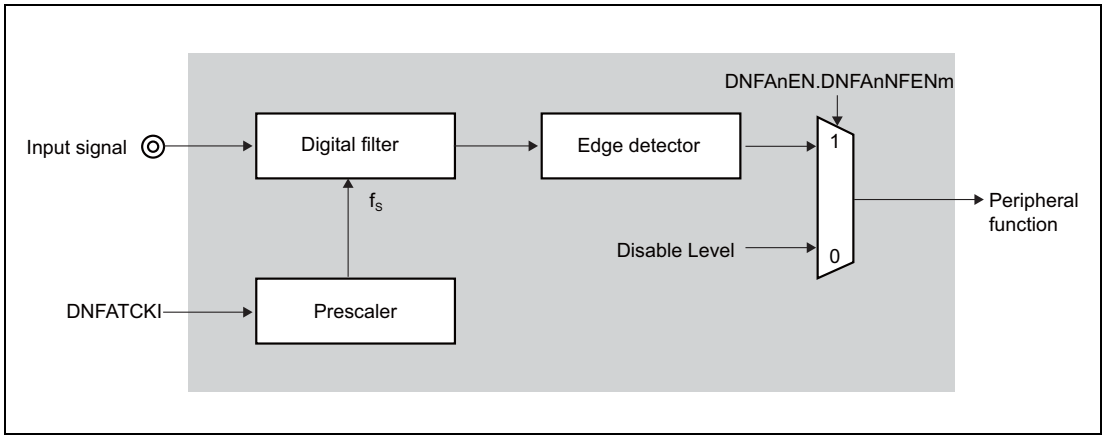



**Figure 2.20 Block diagram of digital filter type C**

The generated signal depends on the register setting, as shown in the following table.

**Table 2.47 Output options for digital filter type C**

| DNFAnEN.DNFAnNFENm | Signals output to peripheral functions |
|---|---|
| 0 | Fixed to disable level |
| 1 | Input signal passed through filter |

### (2) Digital filter type D

A block diagram of digital filter type D is shown below.

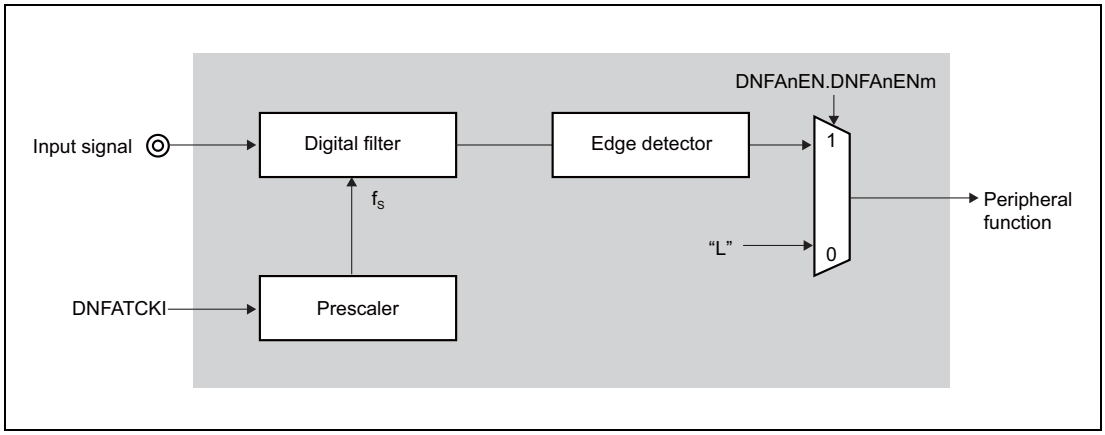



**Figure 2.21 Block diagram of digital filter type D**

The generated signal depends on the register setting, as shown in the following table.

**Table 2.48 Output options for digital filter type D**

| DNFAnEN.DNFAnNFENm | Signals output to peripheral functions |
|---|---|
| 0 | Fixed to disable level |
| 1 | Input signal passed through filter |

### (3) Analog and Digital filter type E

A block diagram of filter type E is shown below.

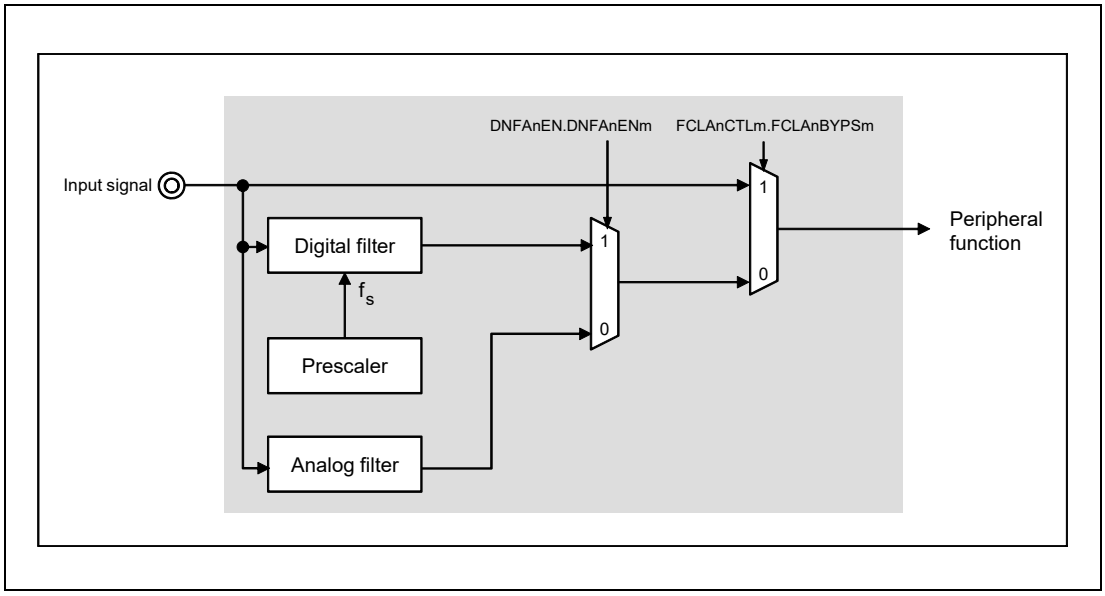



**Figure 2.22 Block diagram of analog and digital filter type E**

The generated signal depends on the register setting, as shown in the following table.

**Table 2.49 Selection of filter type for filter type E**

| DNFAnEN.DNFAnNFENm | Signals output to peripheral functions |
|---|---|
| 0 | Input signal passed through Analog filter |
| 1 | Input signal passed through Digital filter |

**Table 2.50 Filter type E event detection conditions**

| FCLAnBYPSm | FCLAnINTFm | FCLAnINTRm | Edge detection | Bypass |
|---|---|---|---|---|
| 0 | 0 | 0 | No edge detected | Select filter signal |
| | 0 | 1 | Rising edge | |
| | 1 | 0 | Falling edge | |
| | 1 | 1 | Rising and falling edges | |
| 1 | X | X | disable | Select filter-bypass signal |

### 2.7.3.3 Filter control registers

The analog and digital filters are controlled and operated by the following registers:

| Register name | Symbol | Address |
|---|---|---|
| Filter control register m | FCLAnCTLm | See **Table 2.44, Noise Filter Register List** |
| Digital noise elimination control register | DNFAnCTL | |
| Digital noise elimination enable register | DNFAnEN | |
| Digital noise elimination enable L register | DNFAnENL | |

#### (1) FCLAnCTLm — Filter Control Register

This register controls the analog and digital filter operation.

Because the control options for analog and digital filters partially differ, register descriptions are provided separately.

**Access:** This register can be read or written in 8-bit or 1-bit units.

**Address:** See **Table 2.44, Noise Filter Register List**

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FCLAnBYPSm | — | — | — | — | FCLAnINTLm | FCLAnINTFm | FCLAnINTRm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W |

Table 2.51 FCLAnCTLm register contents

| Bit | Bit Name | Function |
|---|---|---|
| 7 | FCLAnBYPSm | Bypass mode selection<br>0: Select filter signal<br>1: Select filter-bypass signal<br>Note: This bit is only valid for filter type E. |
| 2 | FCLAnINTLm | Detection mode selection<br>0: Edge detection<br>1: Level detection<br>Note: This bit is only valid for analog filter type A. |
| 1 | FCLAnINTFm | • In level detection mode (FCLAnINTLm = 1): This bit has no effect.<br>• In edge detection mode (FCLAnINTLm = 0): Falling edge detection control<br>0: Falling edge detection disabled<br>1: Falling edge detection enabled<br>Note: This bit is only valid for filter type A and filter type C.<br>However, digital filter type C is placed in edge detection mode. |
| 0 | FCLAnINTRm | • In level detection mode (FCLAnINTLm = 1): Detected level selection<br>0: Low level detection<br>1: High level detection<br>• In edge detection mode (FCLAnINTLm = 0): Rising edge detection control<br>0: Rising edge detection disabled<br>1: Rising edge detection enabled<br>Note: This bit is only valid for filter type A and filter type C. |

**CAUTION**

Analog filter type A: Be sure to set 0 to bit 7.
Digital filter type C: Be sure to set 0 to bits 7 and 2.

### (2) DNFAnCTL — Digital noise elimination control register

This register is used to specify the filter characteristics of the digital noise elimination filter.

**NOTES**

1. This register is only valid for digital filter type.

**Access:** This register can be read or written in 8-bit or 1-bit units.
**Address:** See **Table 2.44, Noise Filter Register List**
**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | DNFAnNFSTS[1:0] | | — | — | DNFAnPRS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R | R | R/W | R/W | R/W |

**Table 2.52 DNFAnCTL register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 6, 5 | DNFAnNFSTS [1:0] | The DNFAnNFSTS[1:0] bits specify the number of samples used to judge whether an external signal pulse is valid. |

| DNFAnNFSTS[1:0] | Number of samples |
|---|---|
| $00_B$ | 2 |
| $01_B$ | 3 |
| $10_B$ | 4 |
| $11_B$ | 5 |

| Bit | Bit Name | Function |
|---|---|---|
| 2 to 0 | DNFAnPRS[2:0] | Digital filter sampling clock selection |

| DNFAnPRS[2:0] | Sampling clock frequency |
|---|---|
| $000_B$ | DNFATCKI/1 |
| $001_B$ | DNFATCKI/2 |
| $010_B$ | DNFATCKI/4 |
| $011_B$ | DNFATCKI/8 |
| $100_B$ | DNFATCKI/16 |
| $101_B$ | DNFATCKI/32 |
| $110_B$ | DNFATCKI/64 |
| $111_B$ | DNFATCKI/128 |

### (3) DNFAnEN — Digital noise elimination enable register

This register enables and disables digital noise elimination for a specified input signal.

**NOTE**

This register is only valid for digital filter type.

**Access:** This register can be read or written in 16-bit or 1-bit units.
The lower-order bytes (DNFAnNFENL[7:0]) are accessible in 8- or 1-bit units respectively by setting DNFAnENL.DNFAnNFENL[7:0].

**Address:** See **Table 2.44, Noise Filter Register List**

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DNFAnNFENH7 | DNFAnNFENH6 | DNFAnNFENH5 | DNFAnNFENH4 | DNFAnNFENH3 | DNFAnNFENH2 | DNFAnNFENH1 | DNFAnNFENH0 | DNFAnNFENL7 | DNFAnNFENL6 | DNFAnNFENL5 | DNFAnNFENL4 | DNFAnNFENL3 | DNFAnNFENL2 | DNFAnNFENL1 | DNFAnNFENL0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 2.53 DNFAnEN register contents**

| Bit | Bit Name | Function |
|---|---|---|
| 15 to 0 | DNFAnNFENH [7:0], DNFAnNFENL [7:0] | Digital noise elimination control<br>0: Digital noise elimination disabled<br>1: Digital noise elimination enabled |

### (4) DNFAnENL — Digital noise elimination enable L register

Setting in this register correspond to those of the 8 lower-order bits of the DNFAnEN register..

**NOTE**

This register is only valid for digital filter type.

**Access:** This register can be read or written in 8-bit or 1-bit units.

**Address:** See **Table 2.44, Noise Filter Register List**

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DNFAnNFENL7 | DNFAnNFENL6 | DNFAnNFENL5 | DNFAnNFENL4 | DNFAnNFENL3 | DNFAnNFENL2 | DNFAnNFENL1 | DNFAnNFENL0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

For details of the respective bit functions, see **2.7.3.3 (3), DNFAnEN — Digital noise elimination enable register**

## 2.8 Difference among P1M-C, P1H-C and P1H-CE

Table 2.54 Related Differences

| Differences | Describing Section |
|---|---|
| Pin Connection Diagrams | **Section 2.1** |
| Pin Assignments | **Section 2.2 Table 2.1, Pin List** |
| Function Assignments | **Section 2.2 Table 2.3, Pin Function assignments** |

# Section 3 CPU System

## 3.1 Overview

### 3.1.1 Block Configuration

**Figure 3.1** shows the block configuration diagram.

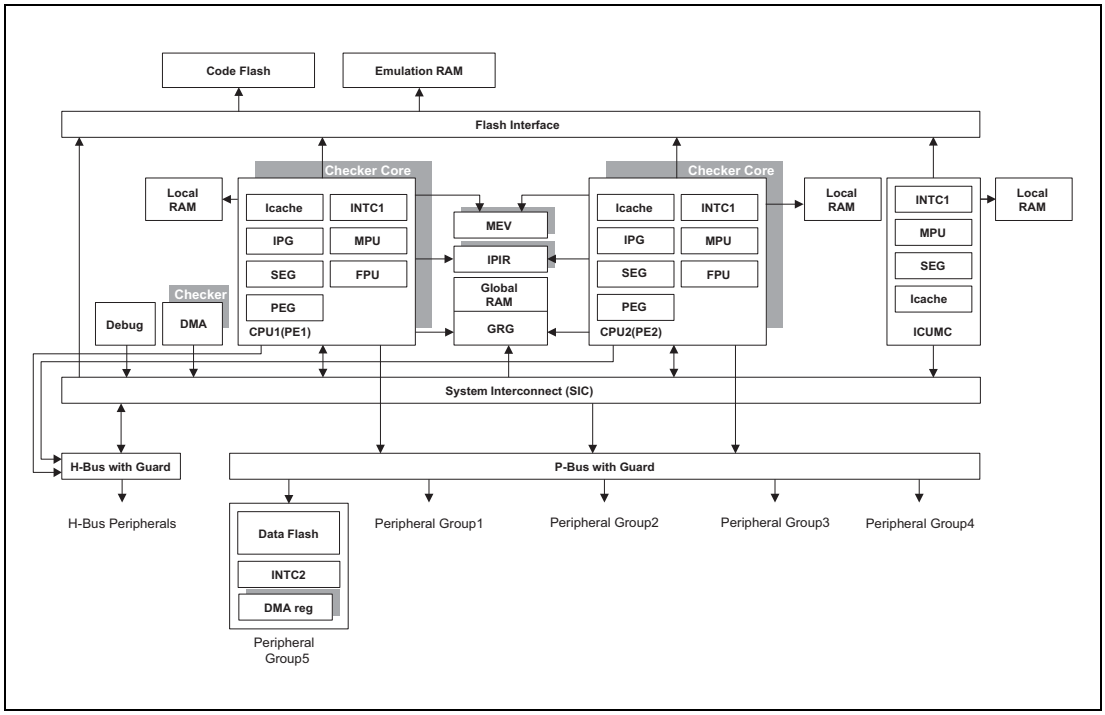



**Figure 3.1 Block Configuration Diagram (maximum configuration of P1H-CE)**

#### CPU1 (PE1), CPU2 (PE2)

The RH850 G3M Core is included as a main CPU. CPUs also include the Checker Core for safety assurance.

#### ICUMC

See *RH850/P1x-C User's Manual: Hardware (Security)* ***Section 3 ICUMC***.

#### Local RAM

Each PE has a high-speed accessible Local RAM.

#### Global RAM

The Global RAM is a mass on-chip RAM that all PEs can share.

#### Code Flash

A mass Code Flash is included for program storage. CPU1, CPU2 and ICUMC share the Code Flash and they are connected via the Code Flash Interface.

### Emulation RAM

It is a RAM to emulate the Code Flash. The programs can be replaced by an external tool without rewriting the Code Flash.

### Data Flash

It is a flash memory being rewritable by the CPU.

### P-Bus and H-Bus

Peripherals are connected on P-Bus and H-Bus. The P-Bus consists of 5 groups, Peripheral Group 1 to 5.

### INTC1, INTC2

INTC1 is an interrupt controller exclusive to each PE. INTC2 is a common interrupt controller that all PEs can share, being able to set the binding destination PE of an interrupt request by the registers.

### DMA

Two DMA transfer modules, DMAC and DTS, are included. DMA includes the checker logic for safety assurance.

### Slave Guard

The slave guard is a system to prevent unauthorized access from the specific bus master, consisting of the following guard structures:

(1) PE Guard (PEG)
The PE guard is a system to prevent unauthorized access to the resources (Local RAM) in the PE from the external master. Access from the PE itself is only enabled but all other accesses except ICUMC are disabled after release from the reset state. ICUMC can read Local RAM of PEs after release from the reset state but cannot write.

(2) Internal Peripheral Guard (IPG)
Each PE has an “Internal Peripheral Guard” (IPG) that protects the registers of peripherals inside the PE against invalid accesses.

(3) System Error Generation (SEG)
The registers controls how to response SYSERR.

(4) Global RAM Guard (GRG)
The global RAM guard is a system to prevent unauthorized access to the global RAM from the external master. The global RAM is unprotected (accessible from all bus masters) after release from the reset state.

### IPIR

The IPIR handles inter PE interrupt, specific to multicore configuration.

This handles inter processor interrupt generation.

### MEV

MEV is Mutual Exclusion Variable Register, specific to multicore configuration.

The CAXI instruction can be used on these registers to implement semaphores.

# 3.2 CPU

## 3.2.1 Core Functions

The Renesas 32-bit RISC architecture RH850 define hardware virtualization as architecture features, but these features by P1x-C Group is not supported, although RH850 documents including this document hereafter may describe hardware threading or virtual CPUs. “Software manual” mentioned in this document means “RH850G3M User's Manual: Software”.

### 3.2.1.1 Features

**Table 3.1** lists features of the RH850G3M core.

**Table 3.1 Features of the RH850G3M Core**

| Item | Feature |
|---|---|
| CPU | • Advanced 32-bit architecture for embedded control<br>• 32 32-bit general registers<br>• RISC-type instruction sets<br>– Long-/short-format load/store instructions<br>– Three-operand instructions<br>• CPU operating modes<br>– User mode, supervisor mode<br>• Address space: 4-Gbyte linear address space for both data and instructions<br>• Instructions: A snooze instruction (SNOOZE) is included for temporary suspension by switching the CPU clock signal off for 32 clock cycles. |
| Coprocessor | • A floating-point operation coprocessor (FPU) mounted<br>– Supports single precision (32 bits) and double precision (64 bits).<br>– Supports data types and exceptions conforming to IEEE754.<br>– Rounding mode: Neighborhood, 0 direction, +∞ direction, and -∞ direction<br>– Handling denormalized numbers: Rounding down to 0 or exception notification to conform to IEEE754 |
| Exception/ Interrupt | • 16 interrupt priority levels settable for each channel<br>• Vector selection method selectable according to performance request or memory usage<br>– Direct branching exception vectors<br>– Indirect branching exception vectors referring to the address table<br>• Supports the high-speed save/restore processing of the context by the dedicated instructions (PUSHSP and POPSP) at the generation of an interrupt |
| Memory Management | • Memory protection function (MPU): 16 areas settable |
| Cache | • Instruction cache |

### 3.2.1.2 Register Set

#### (1) Program Registers

Program registers include the general-purpose registers (r0 to r31) and program counter (PC).

Table 3.2 List of Program Registers

| Program Register | Name | Function | Description |
|---|---|---|---|
| General-purpose registers | r0 | Zero register | Always retains “0” |
| | r1 | Assembler reserved register | Used as working register for generating addresses |
| | r2 | Register for address and data variables (used when the real-time OS used does not use this register) | |
| | r3 | Stack pointer (SP) | Used for generating a stack frame when a function is called |
| | r4 | Global pointer (GP) | Used for accessing a global variable in the data area |
| | r5 | Text pointer (TP) | Used as a register that indicates the start of the text area (area where program code is placed) |
| | r6 to r29 | Register for addresses and data variables | |
| | r30 | Element pointer (EP) | Used as a base pointer for generating addresses when accessing memory |
| | r31 | Link pointer (LP) | Used when the compiler calls a function |
| Program counter | PC | Retains instruction addresses during execution of programs | |

**NOTE**

For further descriptions of r1, r3 to r5, and r31 used for an assembler and/or C compiler, see the specification of each software development environment.

##### (a) General-Purpose Registers

A total of 32 general-purpose registers (r0 to r31) are provided. All of these registers can be used for either data variables or address variables.

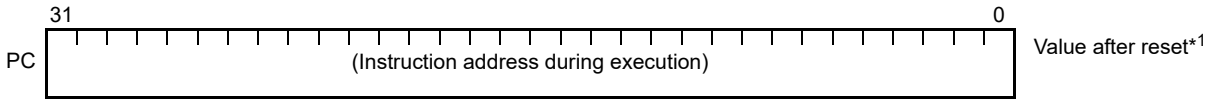
Of the general-purpose registers, r0 to r5, r30, and r31 are assumed to be used for special purposes in software development environments, so it is necessary to note the following when using them.

1. r0, r3, r30
   These registers are implicitly used by instructions.
   r0 is a register that always retains “0”. It is used for operations that use 0 and addressing with base address being 0.
   r3 is implicitly used by the PREPARE, DISPOSE, PUSHSP, and POPSP instructions.
   r30 is used as a base pointer when the SLD instruction or SST instruction accesses memory.
2. r1, r4, r5, r31
   These registers are implicitly used by the assembler and C compiler.
   When using these registers, register contents must first be saved so they are not lost and can be restored after the registers are used.

3. r2
   This register might be used by a real-time OS in some cases. If the real-time OS that is being used is not using r2, r2 can be used as a register for address variables or data variables.

### (b) PC – Program Counter

The PC retains the address of the instruction being executed. Bit 0 is fixed to 0, and branching to an odd number address is disabled.




Note 1. depends on RBASE

### (2) Basic System Registers

The basic system registers are used to control CPU status and to retain exception information.

System registers are read from or written to by using the LDSR and STSR instructions and specifying the system register number, which is made up of a register number and selection ID.

**Table 3.3 Basic System Registers**

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR0, 0 | EIPC | Status save registers when acknowledging EI level exception | SV |
| SR1, 0 | EIPSW | Status save registers when acknowledging EI level exception | SV |
| SR2, 0 | FEPC | Status save registers when acknowledging FE level exception | SV |
| SR3, 0 | FEPSW | Status save registers when acknowledging FE level exception | SV |
| SR 5, 0 | PSW | Program status word | *1 |
| SR 6, 0 | FPSR | Refer to FPU function registers. | CU0 and SV |
| SR 7, 0 | FPEPC | Refer to FPU function registers. | CU0 and SV |
| SR8, 0 | FPST | Refer to FPU function registers. | CU0 |
| SR9, 0 | FPCC | Refer to FPU function registers. | CU0 |
| SR10, 0 | FPCFG | Refer to FPU function registers. | CU0 |
| SR 11, 0 | FPEC | Refer to FPU function registers. | CU0 and SV |
| SR 13, 0 | EIIC | EI level exception cause | SV |
| SR 14, 0 | FEIC | FE level exception cause | SV |
| SR 16, 0 | CTPC | CALLT execution status save register | UM |
| SR 17, 0 | CTPSW | CALLT execution status save register | UM |
| SR 20, 0 | CTBP | CALLT base pointer | UM |
| SR 28, 0 | EIWR | EI level exception working register | SV |
| SR 29, 0 | FEWR | FE level exception working register | SV |
| SR 31, 0 | BSEL | Not implemented. A value of 0 is returned when read and writing is ignored. | SV |
| SR0, 1 | MCFG0 | Machine configuration | SV |
| SR2, 1 | RBASE | Reset vector base address | SV |
| SR3, 1 | EBASE | Exception handler vector address | SV |
| SR4, 1 | INTBP | Base address of the interrupt handler table | SV |
| SR 5, 1 | MCTL | CPU control | SV |
| SR 6, 1 | PID | Processor ID | SV |
| SR 11, 1 | SCCFG | SYSCALL operation setting | SV |
| SR 12, 1 | SCBP | SYSCALL base pointer | SV |
| SR0, 2 | HTCFG0 | Thread configuration | SV |
| SR6, 2 | MEA | Memory error address | SV |
| SR7, 2 | ASID | Address space ID | SV |
| SR8, 2 | MEI | Memory error information | SV |

Note 1. The access permission differs depending on the bit.

(a) EIPC —Status save register when acknowledging EI level exception

When an EI level exception is acknowledged, the address of the instruction that was being executed when the EI level exception occurred, or of the next instruction, is saved to the EIPC register (see **5.1.2 Types of exceptions in Software Manual**).

Because there is only one pair of EI level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

Be sure to set an even-numbered address to the EIPC register. An odd-numbered address must not be specified.

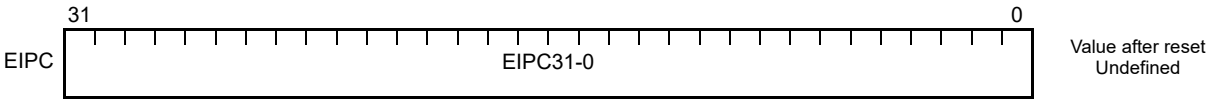



**Table 3.4 EIPC Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 1 | EIPC31-1 | These bits indicate the PC saved when an EI level exception is acknowledged. | R/W | Undefined |
| 0 | EIPC0 | This bit indicates the PC saved when an EI level exception is acknowledged. Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the EIRET instruction is executed is 0. | R/W | Undefined |

(b) EIPSW — Status save register when acknowledging EI level exception

When an EI level exception is acknowledged, the current PSW setting is saved to the EIPSW register.

Because there is only one pair of EI level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

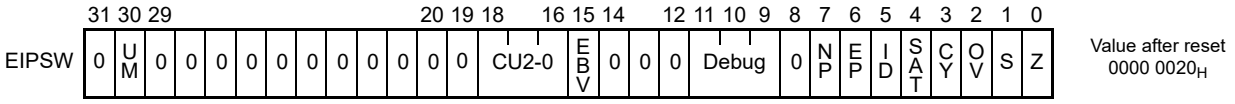



**Table 3.5 EIPSW Register Contents (1/2)**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 30 | UM | This bit stores the PSW.UM bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 29 to 19 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 18 to 16 | CU2-0 | These bits store the PSW.CU 2 to 0 field setting when an EI level exception is acknowledged. | R/W | 0 |
| 15 | EBV | This bit stores the PSW.EBV bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 14 to 12 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 11 to 9 | Debug | These are bits for debugging. | R/W | 0 |
| 8 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 7 | NP | This bit stores the PSW.NP bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 6 | EP | This bit stores the PSW.EP bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 5 | ID | This bit stores the PSW.ID bit setting when an EI level exception is acknowledged. | R/W | 1 |
| 4 | SAT | This bit stores the PSW.SAT bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 3 | CY | This bit stores the PSW.CY bit setting when an EI level exception is acknowledged. | R/W | 0 |

Table 3.5 EIPSW Register Contents (2/2)

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 2 | OV | This bit stores the PSW.OV bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 1 | S | This bit stores the PSW.S bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 0 | Z | This bit stores the PSW.Z bit setting when an EI level exception is acknowledged. | R/W | 0 |

(c) FEPC —Status save register when acknowledging FE level exception

When an FE level exception is acknowledged, the address of the instruction that was being executed when the FE level exception occurred, or of the next instruction, is saved to the FEPC register (see **5.1.2 Types of exceptions in Software Manual**).

Because there is only one pair of FE level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

Be sure to set an even-numbered address to the FEPC register. An odd-numbered address must not be specified.

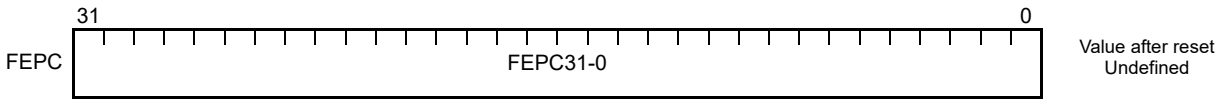



Table 3.6 FEPC Register Contents

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 1 | FEPC31-1 | These bits indicate the PC saved when an FE level exception is acknowledged. | R/W | Undefined |
| 0 | FEPC0 | This bit indicates the PC saved when an FE level exception is acknowledged. Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the FERET instruction is executed is 0. | R/W | Undefined |

(d) FEPSW — Status save register when acknowledging FE level exception

When an FE level exception is acknowledged, the current PSW setting is saved to the FEPSW register.

Because there is only one pair of FE level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

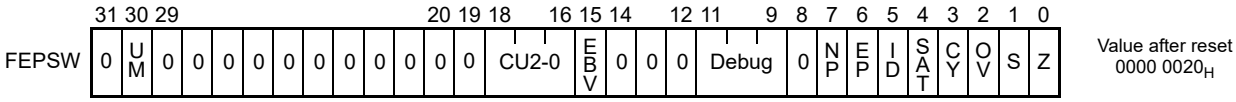



**Table 3.7 FEPSW Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 | — | (Reserved for future expansion. Be sure to clear to 0.) | R/W | 0 |
| 30 | UM | This bit stores the PSW.UM bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 29 to 19 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 18 to 16 | CU2-0 | These bits store the PSW.CU 2 to 0 field setting when an FE level exception is acknowledged. | R/W | 0 |
| 15 | EBV | This bit stores the PSW.EBV bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 14 to 12 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 11 to 9 | Debug | These are bits for debugging. | R/W | 0 |
| 8 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 7 | NP | This bit stores the PSW.NP bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 6 | EP | This bit stores the PSW.EP bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 5 | ID | This bit stores the PSW.ID bit setting when an FE level exception is acknowledged. | R/W | 1 |
| 4 | SAT | This bit stores the PSW.SAT bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 3 | CY | This bit stores the PSW.CY bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 2 | OV | This bit stores the PSW.OV bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 1 | S | This bit stores the PSW.S bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 0 | Z | This bit stores the PSW.Z bit setting when an FE level exception is acknowledged. | R/W | 0 |

(e) PSW — Program status word

PSW (program status word) is a set of flags that indicate the program status (instruction execution result) and bits that indicate the operation status of the CPU (flags are bits in the PSW that are referenced by a condition instruction (Bcond, CMOV, etc.)).

When the LDSR instruction is used to change the contents of each bit in this register, the changed contents become valid immediately after completion of an LDSR instruction execution.

The access permission for the PSW register differs depending on the bit. All bits can be read, but some bits can only be written under certain conditions. See **Table 3.8, Access Permission for PSW Register** for the access permission for each bit.

**Table 3.8 Access Permission for PSW Register**

| Bit | | Access Permission When Reading | Access Permission When Writing | Supplement |
|---|---|---|---|---|
| 30 | UM | UM | SV*1 | |
| 18 to 16 | CU2-0 | | SV*1 | |
| 15 | EBV | | SV*1 | |
| 11 to 9 | Debug | | Special*1 | |
| 7 | NP | | SV*1 | |
| 6 | EP | | SV*1 | |
| 5 | ID | | SV*1 | |
| 4 | SAT | | UM | |
| 3 | CY | | UM | |
| 2 | OV | | UM | |
| 1 | S | | UM | |
| 0 | Z | | UM | |

Note 1. The access permission for the whole PSW register is UM, so the PIE exception does not occur even if the register is written by using an LDSR instruction when PSW.UM is 1. In this case, writing is ignored.

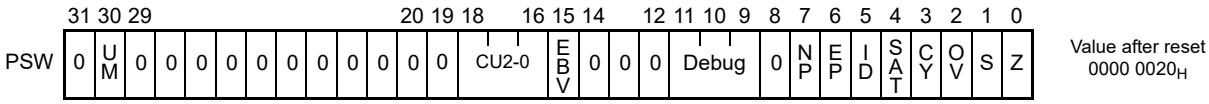



**Table 3.9 PSW Register Contents (1/2)**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 30 | UM | This bit indicates that the CPU is in user mode (UM mode)<br>0: Supervisor mode<br>1: User mode | R/W | 0 |
| 29 to 19 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 18, 17 | CU2-CU1 | These bits indicate the coprocessor use permissions.<br>(Reserved for future expansion. Be sure to clear to 0.) | R | 00 |
| 16 | CU0 | This bit indicates the coprocessor use permissions.<br>When the bit corresponding to the coprocessor is 0, a coprocessor use prohibition exception is generated in response to execution of a coprocessor instruction or access to coprocessor resources (system registers).<br>CU0 bit 16: FPU | R/W | 0 |
| 15 | EBV | This bit indicates the reset vector and exception vector operation. See the descriptions of the RBASE register in **(q), RBASE — Reset vector base address register** and the EBASE register in **(r), EBASE — Exception handler vector address register**. | R/W | 0 |
| 14 to 12 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 11 to 9 | Debug | These bits are used in the debugging functions of development tools. In normal operation, clear these bits to 0. | R/W | 0 |
| 8 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 7 | NP | This bit indicates that an FE-level exception is being processed. When an FE level exception is acknowledged, this bit is set to 1, which prohibits occurrence of multiple exceptions.<br>0: FE level exception handling is not in progress.<br>1: FE level exception handling is in progress. | R/W | 0 |
| 6 | EP | This bit indicates that an exception other than an interrupt is being serviced. It is set to 1 when the corresponding exception occurs. This bit does not affect acknowledging an exception request even when it is set to 1.<br>0: An interrupt is being serviced.<br>1: An exception other than an interrupt is being serviced. | R/W | 0 |
| 5 | ID | This bit indicates that an EI-level exception can be acknowledged.<br>It is used to disable EI level exceptions from being acknowledged during critical sections of an ordinary program or while a EI level interrupt is being serviced.<br>It is set to 1 when the DI instruction is executed, and cleared to 0 when the EI instruction is executed.<br>When an EI level exception is acknowledged, this bit is set to 1 by hardware to disable generation of multiple interrupts by default.<br>0: Acknowledgement of EI level exception is enabled (after execution of EI instruction).<br>1: Acknowledgement of EI level exception is disabled (after execution of DI instruction or after acknowledgement of an EI level exception). | R/W | 1 |
| 4 | SAT*1 | This bit indicates that the operation result is saturated because the result of a saturated operation instruction operation has overflowed. This is a cumulative flag, so when the operation result of the saturated operation instruction becomes saturated, this bit is set to 1, but it is not cleared to 0 when the operation result for a subsequent instruction is not saturated. This bit is cleared to 0 by the LDSR instruction. This bit is neither set to 1 nor cleared to 0 when an arithmetic operation instruction is executed.<br>0: Not saturated<br>1: Saturated | R/W | 0 |

Table 3.9 PSW Register Contents (2/2)

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 3 | CY | This bit indicates whether a carry or borrow has occurred in the operation result.<br>0: Carry and borrow have not occurred.<br>1: Carry or borrow has occurred. | R/W | 0 |
| 2 | OV*1 | This bit indicates whether or not an overflow has occurred during an operation.<br>0: Overflow has not occurred.<br>1: Overflow has occurred. | R/W | 0 |
| 1 | S*1 | This bit indicates whether or not the result of an operation is negative.<br>0: Result of operation is positive or 0.<br>1: Result of operation is negative | R/W | 0 |
| 0 | Z | This bit indicates whether or not the result of an operation is 0.<br>0: Result of operation is not 0.<br>1: Result of operation is 0. | R/W | 0 |

Note 1. The operation result of the saturation processing is determined in accordance with the contents of the OV flag and S flag during a saturated operation. When only the OV flag is set to 1 during a saturated operation, the SAT flag is set to 1.

| Operation result status | Flag status | | | Operation result after saturation processing |
|---|---|---|---|---|
| | SAT | OV | S | |
| Exceeded positive maximum value | 1 | 1 | 0 | 7FFF FFFF$_H$ |
| Exceeded negative maximum value | 1 | 1 | 1 | 8000 0000$_H$ |
| Positive (maximum value not exceeded) | Value prior to operation is retained. | 0 | 0 | Operation result itself |
| Negative (maximum value not exceeded) | | | 1 | |

### (f) EIIC — EI level exception source register

The EIIC register retains the source of any EI level exception that occurs. The value retained in this register is an exception code corresponding to a specific exception source.

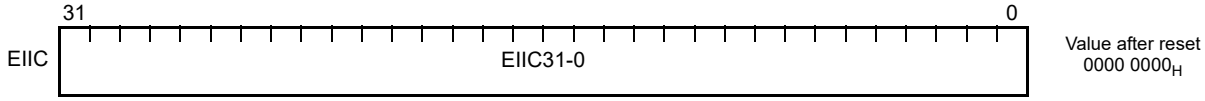



**Table 3.10 EIIC Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 0 | EIIC31-0 | These bits store the exception source code when an EI level exception occurs.<br>The EIIC 15 to 0 field stores the lower 16 bits of the exception source code.<br>The EIIC31 to 16 field stores detailed exception source codes defined individually for each exception. If there is no particular definition, these bits are set to 0. | R/W | 0 |

### (g) FEIC — FE level exception source register

The FEIC register retains the source of any FE level exception that occurs. The value retained in this register is an exception code corresponding to a specific exception source.

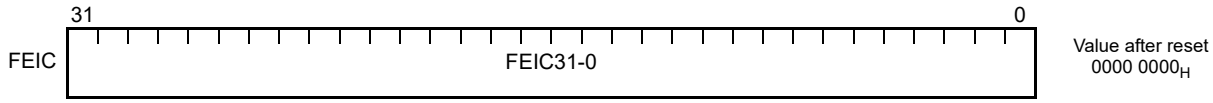



**Table 3.11 FEIC Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 0 | FEIC31-0 | These bits store the exception source code when an FE level exception occurs.<br>The FEIC 15 to 0 field stores the lower 16 bits of the exception source code.<br>The FEIC31 to 16 field stores detailed exception source codes defined individually for each exception. If there is no particular definition, these bits are set to 0. | R/W | 0 |

### (h) CTPC — Status save register when executing CALLT register

When a CALLT instruction is executed, the address of the next instruction after the CALLT instruction is saved to CTPC. Be sure to set an even-numbered address to the CTPC register. An odd-numbered address must not be specified.

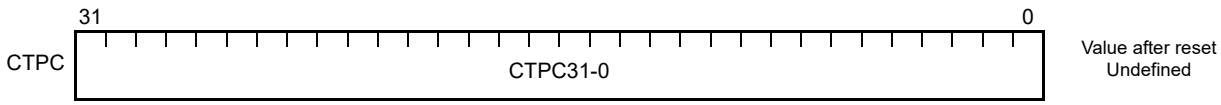



**Table 3.12 CTPC Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 1 | CTPC31-1 | These bits indicate the PC of the instruction after the CALLT instruction. | R/W | Undefined |
| 0 | CTPC0 | This bit indicates the PC of the instruction after the CALLT instruction.<br>Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the CTRET instruction is executed is 0. | R/W | Undefined |

(i) CTPSW — Status save register when executing CALLT register

When a CALLT instruction is executed, some of the PSW (program status word) settings are saved to CTPSW.

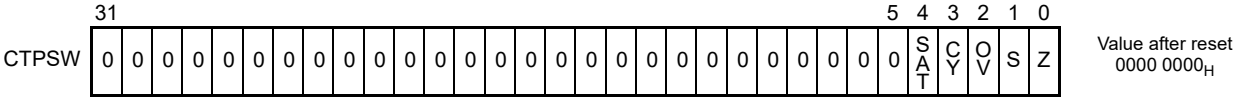



Table 3.13 CTPSW Register Contents

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 5 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 4 | SAT | This bit stores the PSW.SAT bit setting when the CALLT instruction is executed. | R/W | 0 |
| 3 | CY | This bit stores the PSW.CY bit setting when the CALLT instruction is executed. | R/W | 0 |
| 2 | OV | This bit stores the PSW.OV bit setting when the CALLT instruction is executed. | R/W | 0 |
| 1 | S | This bit stores the PSW.S bit setting when the CALLT instruction is executed. | R/W | 0 |
| 0 | Z | This bit stores the PSW.Z bit setting when the CALLT instruction is executed. | R/W | 0 |

(j) CTBP — CALLT base pointer register

The CTBP register is used to specify table addresses of the CALLT instruction and generate target addresses. Be sure to set the CTBP register to a halfword address.

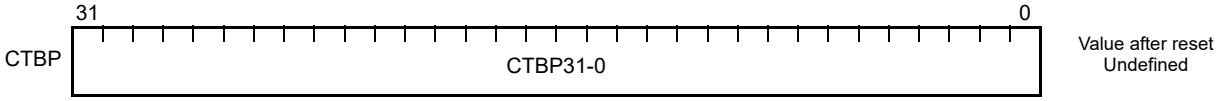



Table 3.14 CTBP Register Contents

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 1 | CTBP31-1 | These bits indicate the base pointer address of the CALLT instruction.<br>These bits indicate the first address in the table used by the CALLT instruction. | R/W | Undefined |
| 0 | CTBP0 | These bits indicate the base pointer address of the CALLT instruction.<br>These bits indicate the first address in the table used by the CALLT instruction.<br>Always set this bit to 0. | R | 0 |

(k) ASID — Address space ID register

This is the address space ID. This is used to identify the address space provided by the memory management function.

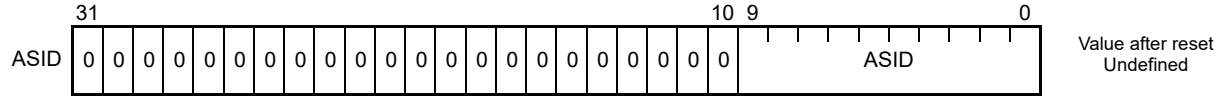



Table 3.15 ASID Register Contents

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 10 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 9 to 0 | ASID | This is the address space ID. | R/W | Undefined |

(l) EIWR — EI level exception working register

The EIWR register is used as a working register when an EI level exception has occurred.

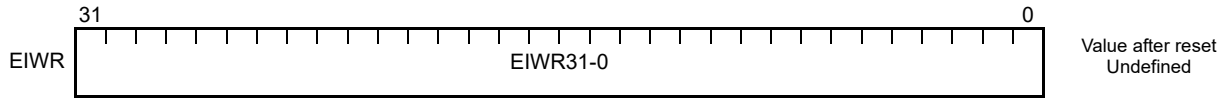



**Table 3.16 EIWR Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 0 | EIWR31-0 | These bits constitute a working register that can be used for any purpose during the processing of an EI level exception. Use this register for purposes such as storing the values of general-purpose registers. | R/W | Undefined |

(m) FEWR — FE level exception working register

The FEWR register is used as a working register when an FE level exception has occurred.

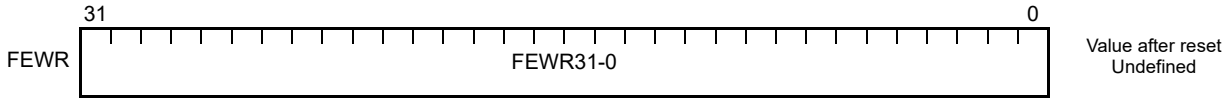



**Table 3.17 FEWR Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 0 | FEWR31-0 | These bits constitute a working register that can be used for any purpose during the processing of an FE level exception. Use this register for purposes such as storing the values of general-purpose registers. | R/W | Undefined |

(n) HTCFG0 — Thread configuration register

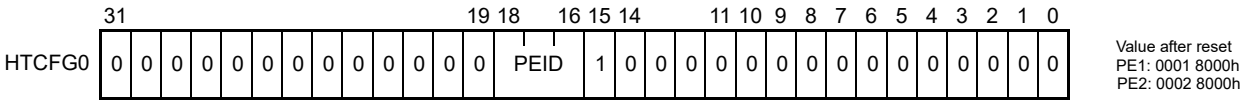



**Table 3.18 HTCFG0 Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 19 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 18 to 16 | PEID | These bits indicate the processor element number. | R | CPU1 (PE1): $001_B$<br>CPU2 (PE2): $010_B$ |
| 15 | — | (Reserved for future expansion. Be sure to set to 1.) | R | 1 |
| 14 to 0 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |

(o) MEA — Memory error address register

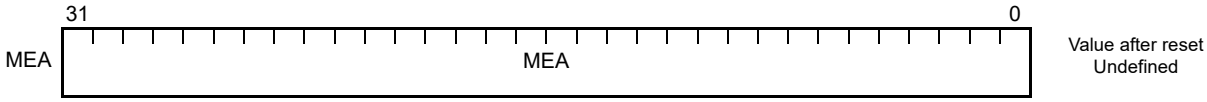



**Table 3.19 MEA Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 0 | MEA | These bits store a virtual address when a MAE (misaligning) or MPU violation occurs. | R/W | Undefined |

(p) MEI — Memory error information register

This register is used to store information about the instruction that caused the exception when a misaligned (MAE) or memory protection (MDP) exception occurs. This information is used during emulation.

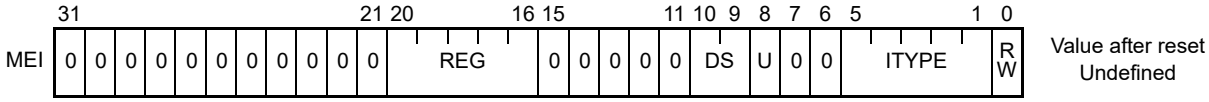



**Table 3.20 MEI Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 21 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 20 to 16 | REG4-0 | These bits indicate the number of the source or destination register accessed by the instruction that caused the exception.<br>For details, see **Table 3.21, Instructions Causing Exceptions and Values of MEI Register**. | R/W | Undefined |
| 15 to 11 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 10, 9 | DS | These bits indicate the type of data handled by the instruction that caused the exception.*1<br>0: Byte (8 bits)<br>1: Halfword (16 bits)<br>2: Word (32 bits)<br>3: Double-word (64 bits)<br>For details, see **Table 3.21, Instructions Causing Exceptions and Values of MEI Register**. | R/W | Undefined |
| 8 | U | These bits indicate the sign extension method of the instruction that caused the exception.<br>0: Signed<br>1: Unsigned<br>For details, see **Table 3.21, Instructions Causing Exceptions and Values of MEI Register**. | R/W | Undefined |
| 7, 6 | – | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 5 to 1 | ITYPE4-0 | These bits indicate the instruction that caused the exception.<br>For details, see **Table 3.21, Instructions Causing Exceptions and Values of MEI Register**. | R/W | Undefined |
| 0 | RW | This bit indicates whether the operation of the instruction that caused the exception was read (Load-memory) or write (Store-memory)<br>0: Read (Load-memory)<br>1: Write (Store-memory)<br>For details, see **Table 3.21, Instructions Causing Exceptions and Values of MEI Register**. | R/W | Undefined |

Note 1. Even if the data is divided and access is made several times due to the specifications of the hardware, the original data type indicated by the instruction is stored.

Table 3.21 Instructions Causing Exceptions and Values of MEI Register

| Instruction | REG | DS | U | RW | ITYPE |
|---|---|---|---|---|---|
| SLD.B | dst | 0 (byte) | 0 (signed) | 0 (read) | 00000b |
| SLD.BU | dst | 0 (byte) | 1 (unsigned) | 0 (read) | 00000b |
| SLD.H | dst | 1 (halfword) | 0 (signed) | 0 (read) | 00000b |
| SLD.HU | dst | 1 (halfword) | 1 (unsigned) | 0 (read) | 00000b |
| SLD.W | dst | 2 (word) | 0 (signed) | 0 (read) | 00000b |
| SST.B | src | 0 (byte) | 0 (signed) | 1 (write) | 00000b |
| SST.H | src | 1 (halfword) | 0 (signed) | 1 (write) | 00000b |
| SST.W | src | 2 (word) | 0 (signed) | 1 (write) | 00000b |
| LD.B (disp16) | dst | 0 (byte) | 0 (signed) | 0 (read) | 00001b |
| LD.BU (disp16) | dst | 0 (byte) | 1 (unsigned) | 0 (read) | 00001b |
| LD.H (disp16) | dst | 1 (halfword) | 0 (signed) | 0 (read) | 00001b |
| LD.HU (disp16) | dst | 1 (halfword) | 1 (unsigned) | 0 (read) | 00001b |
| LD.W (disp16) | dst | 2 (word) | 0 (signed) | 0 (read) | 00001b |
| ST.B (disp16) | src | 0 (byte) | 0 (signed) | 1 (write) | 00001b |
| ST.H (disp16) | src | 1 (halfword) | 0 (signed) | 1 (write) | 00001b |
| ST.W (disp16) | src | 2 (word) | 0 (signed) | 1 (write) | 00001b |
| LD.B (disp23) | dst | 0 (byte) | 0 (signed) | 0 (read) | 00010b |
| LD.BU (disp23) | dst | 0 (byte) | 1 (unsigned) | 0 (read) | 00010b |
| LD.H (disp23) | dst | 1 (halfword) | 0 (signed) | 0 (read) | 00010b |
| LD.HU (disp23) | dst | 1 (halfword) | 1 (unsigned) | 0 (read) | 00010b |
| LD.W (disp23) | dst | 2 (word) | 0 (signed) | 0 (read) | 00010b |
| ST.B (disp23) | src | 0 (byte) | 0 (signed) | 1 (write) | 00010b |
| ST.H (disp23) | src | 1 (halfword) | 0 (signed) | 1 (write) | 00010b |
| ST.W (disp23) | src | 2 (word) | 0 (signed) | 1 (write) | 00010b |
| LD.DW (disp23) | dst | 3 (double-word) | 0 (signed) | 0 (read) | 00010b |
| ST.DW (disp23) | src | 3 (double-word) | 0 (signed) | 1 (write) | 00010b |
| LDL.W | dst | 2 (word) | 0 (signed) | 0 (read) | 00111b |
| STC.W | src | 2 (word) | 0 (signed) | 1 (write) | 00111b |
| CAXI | dst | 2 (word) | 1 (unsigned) | 0 (read)*1 | 01000b |
| SET1 | — | 0 (byte) | 1 (unsigned) | 0 (read)*1 | 01001b |
| CLR1 | — | 0 (byte) | 1 (unsigned) | 0 (read)*1 | 01001b |
| NOT1 | — | 0 (byte) | 1 (unsigned) | 0 (read)*1 | 01001b |
| TST1 | — | 0 (byte) | 1 (unsigned) | 0 (read) | 01001b |
| PREPARE | — | 2 (word) | 1 (unsigned) | 1 (write) | 01100b |
| DISPOSE | — | 2 (word) | 1 (unsigned) | 0 (read) | 01100b |
| PUSHSP | — | 2 (word) | 1 (unsigned) | 1 (write) | 01101b |
| POPSP | — | 2 (word) | 1 (unsigned) | 0 (read) | 01101b |
| SWITCH | — | 1 (halfword) | 0 (signed) | 0 (read) | 10000b |
| CALLT | — | 1 (halfword) | 1 (unsigned) | 0 (read) | 10001b |
| SYSCALL | — | 2 (word) | 1 (unsigned) | 0 (read) | 10010b |
| CACHE | — | — | — | 0/1*2 | 10100b |
| Interrupt (table reference)*3 | — | 2 (word) | 1 (unsigned) | 0 (read) | 10101b |

Note 1. This exception occurs when the instruction executes a read access.

Note 2. It depends on actual operation.

Note 3. An exception occurs when the table reference interrupt vector is read.

**NOTE**

dst: destination register number, src: source register number

(q) RBASE — Reset vector base address register

This register indicates the reset vector address when there is a reset. If the PSW.EBV bit is 0, this vector address is also used as the exception vector address.

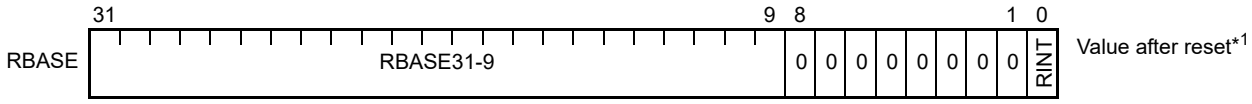



**Table 3.22 RBASE Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 9 | RBASE31-9 | These bits indicate the reset vector when there is a reset. When PSW.EBV = 0, this address is also used as the exception vector. The RBASE8 to 1 bits are not assigned as names because these bits are always 0. | R | |
| 8 to 1 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 0 | RINT | When the RINT bit is set, all maskable interrupts are diverted to one single interrupt handler (Reduced Mode). This bit is valid when PSW.EBV = 0. | R | 0 |

Note 1. Hardware reset will set RBASE31-9 to its predefined value PE1 and PE2 have the same value.

### (r) EBASE — Exception handler vector address register

This register indicates the exception handler vector address. This register is valid when the PSW.EBV bit is 1.

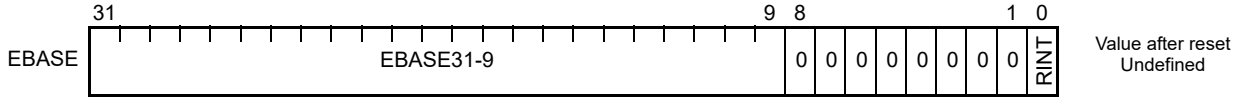



**Table 3.23 EBASE Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 9 | EBASE31-9 | The exception handler routine address is changed to the address resulting from adding the offset address of each exception to the base address specified for this register.<br>The EBASE8 to 1 bits are not assigned as names because these bits are always 0. | R/W | Undefined |
| 8 to 1 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 0 | RINT | When the RINT bit is set, all maskable interrupts are diverted to one single interrupt handler (Reduced Mode). | R/W | Undefined |

### (s) INTBP — Base address of the interrupt handler table register

This register indicates the base address of the address table when the table reference method is selected as the interrupt handler address selection method.

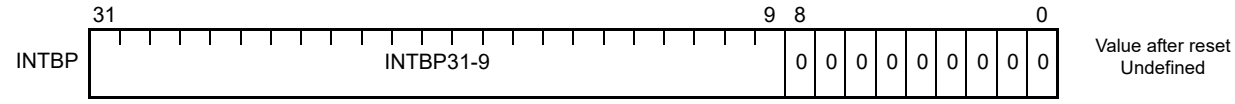



**Table 3.24 INTBP Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 9 | INTBP31-9 | These bits indicate the base pointer address for an interrupt prescribed in the expanded specifications.<br>The value indicated by these bits is the first address in the table used to determine the exception handler when the interrupt prescribed by the expanded specifications (EIINT0 to EIINT511) is acknowledged.<br>The INTBP8 to 0 bits are not assigned as names because these bits are always 0. | R/W | Undefined |
| 8 to 0 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |

(t) PID — Processor ID register

The PID register retains a processor identifier that is unique to the CPU. The PID register is a read-only register.

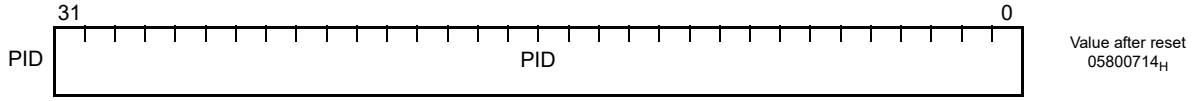



Table 3.25 PID Register Contents

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 24 | PID | Architecture identifier<br>This identifier indicates the architecture of the processor. | R | $05_H$ |
| 23 to 8 | | Function identifier<br>This identifier indicates the functions of the processor.<br>These bits indicate whether or not functions defined per bit are implemented (1: implemented, 0: not implemented).<br>Bit 23-11 Reserved<br>Bit 10 Double-precision floating-point operation function<br>Bit 9 Single-precision floating-point operation function<br>Bit 8 Memory protection unit (MPU) function | R | $8007_H$ |
| 7 to 0 | | Version identifier<br>This identifier indicates the version of the processor. | R | $14_H$ |

(u) SCCFG — SYSCALL operation setting register

This register is used to specify operations related to the SYSCALL instruction. Be sure to set an appropriate value to this register before using the SYSCALL instruction.

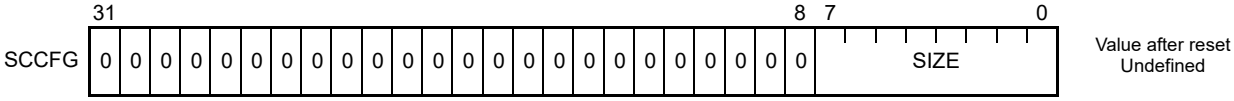



Table 3.26 SCCFG Register Contents

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 8 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 7 to 0 | SIZE | These bits specify the maximum number of entries of a table that the SYSCALL instruction references. The maximum number of entries the SYSCALL instruction references is 1 if SIZE is 0, and 256 if SIZE is 255.<br>By setting the maximum number of entries appropriately in accordance with the number of functions branched by the SYSCALL instruction, the memory area can be effectively used.<br>If a vector exceeding the maximum number of entries is specified for the SYSCALL instruction, the first entry is selected. Place an error processing routine at the first entry. | R/W | Undefined |

(v) SCBP — SYSCALL base pointer register

The SCBP register is used to specify a table address of the SYSCALL instruction and generate a target address. Be sure to set an appropriate value to this register before using the SYSCALL instruction.

Be sure to set a word address to the SCBP register.

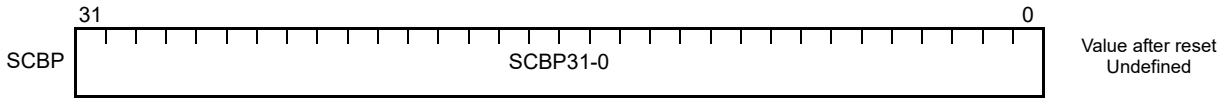



Table 3.27 SCBP Register Contents

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 2 | SCBP31-2 | These bits indicate the base pointer address of the SYSCALL instruction.<br>These bits indicate the first address in the table used by the SYSCALL instruction. | R/W | Undefined |
| 1, 0 | SCBP1-0 | These bits indicate the base pointer address of the SYSCALL instruction.<br>These bits indicate the first address in the table used by the SYSCALL instruction.<br>Always set this bit to 0. | R | 0 |

(w) MCFG0 — Machine configuration register

This register indicates the CPU configuration.

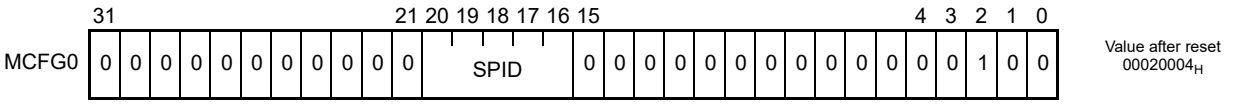



Table 3.28 MCFG0 Register Contents

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 21 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 20 to 16 | SPID | These bits indicate the system protection number. It is not possible to write “$00000_B$”/“$00001_B$” to SPID. If witten, these value is “$00010_B$”. | R/W | $00010_B$ |
| 15 to 3 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 2 | — | (Reserved for future expansion. Be sure to clear to 1.) | R | 1 |
| 1, 0 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |

(x) MCTL — Machine control register

This register is used to control the CPU.

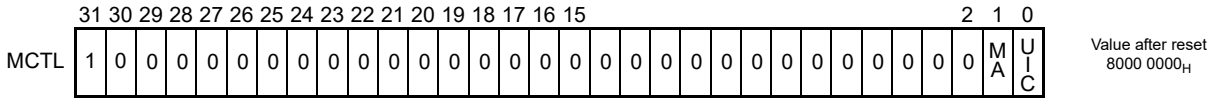



**Table 3.29 MCTL Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 | — | (Reserved for future expansion. Be sure to clear to 1.) | R | 1 |
| 30 to 2 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 1 | MA | This bit specifies the operation when a misaligned access occurs.<br>0: An exception occurs. *1<br>1: Hardware operates normally. | R/W | 0 |
| 0 | UIC | This bit is used to control the interrupt enable/disable operation in user mode. When this bit is set to 1, executing the EI/DI instruction become possible in user mode. | R/W | 0 |

Note 1. Excluding LD.DW and ST.DW instructions executed at an address at a word boundary.

When MA bit is “1”, CPU divides the transaction from misaligned data access to aligned data accesses for 16-bit and 32-bit accesses shown below. In case of this, atomic characteristics are not guaranteed.

In case of 64-bit access, an exception still occurs even when MA bit is “1” for a misaligned access except word boundary address.

| Access Conditions | | Cycles Divided by the CPU | | |
|---|---|---|---|---|
| Data Width | Address | 1st | 2nd | 3rd |
| 16 bits | 2n + 1 | 8-bit access to 2n + 1 | 8-bit access to 2n + 2 | — |
| 32 bits | 4n + 1 | 8-bit access to 4n + 1 | 16-bit access to 4n + 2 | 8-bit access to 4n + 4 |
| 32 bits | 4n + 2 | 16-bit access to 4n + 2 | 16-bit access to 4n + 4 | — |
| 32 bits | 4n + 3 | 8-bit access to 4n + 3 | 16-bit access to 4n + 4 | 8-bit access to 4n + 6 |

### (3) Interrupt Function Registers

Table 3.30 Interrupt Function System Registers

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR7, 1 | FPIPR | FPI exception interrupt priority setting | — |
| SR10, 2 | ISPR | Priority of interrupt being serviced | SV |
| SR11, 2 | PMR | Interrupt priority masking | SV |
| SR12, 2 | ICSR | Interrupt control status | SV |
| SR13, 2 | INTCFG | Interrupt function setting | SV |

(a) FPIPR — FPI exception interrupt priority setting register

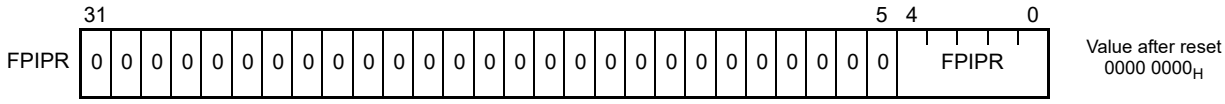



Table 3.31 FPIPR Register Contents

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 5 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 5 to 0 | FPIPR | These bits specify the priority of the floating-point operation exception interrupt (FPI, indicating imprecision). Values from 0 to 16 should be used; the settings 17 and higher are prohibited.<br>FPI exceptions are handled according to this interrupt priority, which is specified in advance. When generated at the same time as another interrupt with the same priority level, the FPI takes priority.<br>NOTE: If these bits are set to 17 or higher values, handling is as if the setting were 16. | R/W | 0 |

(b) ISPR — Priority of interrupt being serviced register

This register holds the priority of the EIINTn interrupt being serviced. This priority value is then used to perform priority ceiling processing when multiple interrupts are generated.

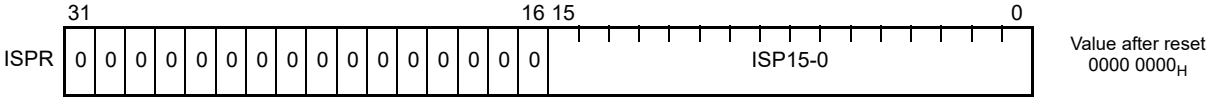



**Table 3.32 ISPR Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 16 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 15 to 0 | ISP15-0 | These bits indicate the acknowledgment status of an EIINTn interrupt with a priority that corresponds to the relevant bit position.<br>0: An interrupt request for an interrupt whose priority corresponds to the relevant bit position has not been acknowledged.<br>1: An interrupt request for an interrupt whose priority corresponds to the relevant position is being serviced by the CPU core.<br><br>The bit positions correspond to the following priority levels:<br><br>**Bit** / **Priority**<br>0 / Priority 0 (highest)<br>1 / Priority 1<br>... <br>14 / Priority 14<br>15 / Priority 15<br><br>When an interrupt request (EIINTn) is acknowledged, the bit corresponding to the acknowledged interrupt request is automatically set to 1. If PSW.EP is 0 when the EIRET instruction is executed, the bit with the highest priority among the ISP15 to 0 bits that are set (0 is the highest priority) is cleared to 0.*1<br>While a bit in this register is set to 1, same and lower priority interrupts (EIINTn) and FPI exception*2 are masked. Priority level judgment is therefore not performed when the system is determining whether to acknowledge an exception, meaning that exceptions will not be acknowledged.<br>When performing software-based priority control using the PMR register, be sure to clear this register by using the INTCFG.ISPC bit. | R*3 | 0 |

Note 1. Interrupt acknowledgment and auto-updating of values when the EIRET instruction is executed are disabled by setting (1) the INTCFG.ISPC bit. It is recommended to enable auto-updating of values, so in normal cases, the INTCFG.ISPC bit should be cleared to 0.

Note 2. Since FPI exceptions have the same level of priority as this interrupt (EIINTn), they are affected by interrupts in the same way as the ISPR. The priority of the FPI exception is set by the FPIPR register.

Note 3. This is R or R/W, depending on the setting of the INTCFG.ISPC bit. It is recommended to use this register as a read-only (R) register.

(c) PMR — Interrupt priority masking register

This register is used to mask the specified interrupt priority.

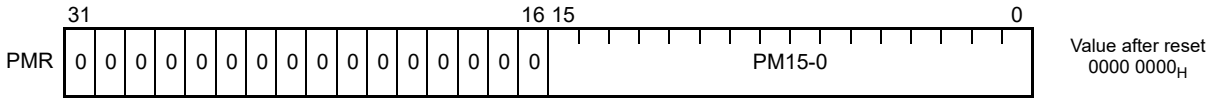



**Table 3.33 PMR Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 16 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 15 to 0 | PM15-0 | These bits mask an interrupt request with a priority level that corresponds to the relevant bit position.<br>0: Servicing of an interrupt request with a priority that corresponds to the relevant bit position is enabled.<br>1: Servicing of an interrupt request with a priority that corresponds to the relevant bit position is disabled.<br><br>The bit positions correspond to the following priority levels:<br><br>**Bit** / **Priority**<br>0 / Priority 0 (highest)<br>1 / Priority 1<br>... <br>14 / Priority 14<br>15 / Priority 15 and priority 16 (lowest)<br><br>While a bit in this register is set to 1, interrupts (EIINTn) with the priority corresponding to that bit and FPI exception*1 are masked. Priority level judgment is therefore not performed when the system is determining whether to acknowledge an exception, meaning that exceptions will not be acknowledged*2. | R/W | 0 |

Note 1. Since a FPI exception is specified as the same level of priority as that of an interrupt (EIINTn), it is affected by the PMR like interrupts. The priority of FPI exception is set by the FPIPR register.

Note 2. Specify the masks by setting the bits to 1 in order from the lowest-priority bit. For example, $FF00_H$ can be set, but $F0F0_H$ or $00FF_H$ cannot.

(d) ICSR — Interrupt control status register

This register indicates the interrupt control status in the CPU.

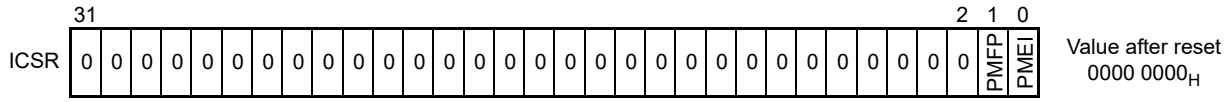



**Table 3.34 ICSR Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 2 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 1 | PMFP | This bit indicates that an FPI exception with the priority level masked by the PMR register exists. Be sure to clear this bit to 0. | R | 0 |
| 0 | PMEI | This bit indicates that an interrupt (EIINTn) with the priority level masked by the PMR register exists | R | 0 |

(e) INTCFG — Interrupt function setting register

This register is used to specify settings related to the CPU's internal interrupt function.

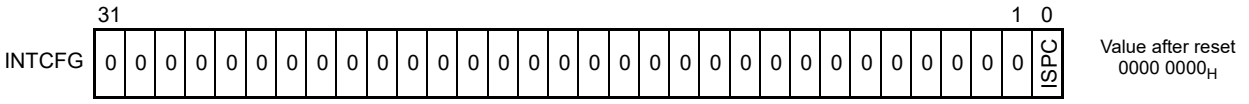



**Table 3.35 INTCFG Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 1 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 0 | ISPC | This bit changes how the ISPR register is written.<br>0: The ISPR register is automatically updated. Updates triggered by the program (via execution of LDSR instruction) are ignored.<br>1: The ISPR register is not automatically updated. Updates triggered by the program (via execution of LDSR instruction) are performed.<br>If this bit is cleared to 0, the bits of the ISPR register are automatically set to 1 when an interrupt (EIINTn) is acknowledged, and cleared to 0 when the EIRET instruction is executed. In this case, the bits are not updated by an LDSR instruction executed by the program.<br>If this bit is set to 1, the bits of the ISPR register are not updated by the acknowledgement of an interrupt (EIINTn) or by execution of the EIRET instruction. In this case, the bits can be updated by an LDSR instruction executed by the program.<br>In normal cases, the ISPC bit should be cleared. When performing software-based priority control, however, set this bit to 1 and perform priority control by using the PMR register. | R/W | 0 |

### (4) FPU Function Registers

The FPU can use the following system registers to control floating-point operation. For details of the registers, see the Software Manual.

Table 3.36 FPU System Registers

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR6, 0 | FPSR | Floating-point operation setting/status | CU0 and SV |
| SR7, 0 | FPEPC | Floating-point operation exception program counter | CU0 and SV |
| SR8, 0 | FPST | Floating-point status | CU0 |
| SR9, 0 | FPCC | Floating-point operation comparison result | CU0 |
| SR10, 0 | FPCFG | Floating-point operation configuration | CU0 |
| SR11, 0 | FPEC | Floating-point operation exception control | CU0 and SV |

### (5) MPU function resisters

Table 3.37 MPU Function System Registers (1/2)

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR0, 5 | MPM | Memory protection operation mode setting | SV |
| SR1, 5 | MPRC | MPU region control | SV |
| SR4, 5 | MPBRGN | MPU base region number | SV |
| SR5, 5 | MPTRGN | MPU end region number | SV |
| SR8, 5 | MCA | Memory protection setting check address | SV |
| SR9, 5 | MCS | Memory protection setting check size | SV |
| SR10, 5 | MCC | Memory protection setting check command | SV |
| SR11, 5 | MCR | Memory protection setting check result | SV |
| SR0, 6 | MPLA0 | Protection area minimum address | SV |
| SR1, 6 | MPUA0 | Protection area maximum address | SV |
| SR2, 6 | MPAT0 | Protection area attribute | SV |
| SR4, 6 | MPLA1 | Protection area minimum address | SV |
| SR5, 6 | MPUA1 | Protection area maximum address | SV |
| SR6, 6 | MPAT1 | Protection area attribute | SV |
| SR8, 6 | MPLA2 | Lower address of the protection area | SV |
| SR9, 6 | MPUA2 | Protection area maximum address | SV |
| SR10, 6 | MPAT2 | Protection area attribute | SV |
| SR12, 6 | MPLA3 | Protection area minimum address | SV |
| SR13, 6 | MPUA3 | Protection area maximum address | SV |
| SR14, 6 | MPAT3 | Protection area attribute | SV |
| SR16, 6 | MPLA4 | Protection area minimum address | SV |
| SR17, 6 | MPUA4 | Protection area maximum address | SV |
| SR18, 6 | MPAT4 | Protection area attribute | SV |
| SR20, 6 | MPLA5 | Protection area minimum address | SV |
| SR21, 6 | MPUA5 | Protection area maximum address | SV |
| SR22, 6 | MPAT5 | Protection area attribute | SV |
| SR24, 6 | MPLA6 | Protection area minimum address | SV |

Table 3.37 MPU Function System Registers (2/2)

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR25, 6 | MPUA6 | Protection area maximum address | SV |
| SR26, 6 | MPAT6 | Protection area attribute | SV |
| SR28, 6 | MPLA7 | Protection area minimum address | SV |
| SR29, 6 | MPUA7 | Protection area maximum address | SV |
| SR30, 6 | MPAT7 | Protection area attribute | SV |
| SR0, 7 | MPLA8 | Protection area minimum address | SV |
| SR1, 7 | MPUA8 | Protection area maximum address | SV |
| SR2, 7 | MPAT8 | Protection area attribute | SV |
| SR4, 7 | MPLA9 | Protection area minimum address | SV |
| SR5, 7 | MPUA9 | Protection area maximum address | SV |
| SR6, 7 | MPAT9 | Protection area attribute | SV |
| SR8, 7 | MPLA10 | Protection area minimum address | SV |
| SR9, 7 | MPUA10 | Protection area maximum address | SV |
| SR10, 7 | MPAT10 | Protection area attribute | SV |
| SR12, 7 | MPLA11 | Protection area minimum address | SV |
| SR13, 7 | MPUA11 | Protection area maximum address | SV |
| SR14, 7 | MPAT11 | Protection area attribute | SV |
| SR16, 7 | MPLA12 | Protection area minimum address | SV |
| SR17, 7 | MPUA12 | Protection area maximum address | SV |
| SR18, 7 | MPAT12 | Protection area attribute | SV |
| SR20, 7 | MPLA13 | Protection area minimum address | SV |
| SR21, 7 | MPUA13 | Protection area maximum address | SV |
| SR22, 7 | MPAT13 | Protection area attribute | SV |
| SR24, 7 | MPLA14 | Protection area minimum address | SV |
| SR25, 7 | MPUA14 | Protection area maximum address | SV |
| SR26, 7 | MPAT14 | Protection area attribute | SV |
| SR28, 7 | MPLA15 | Protection area minimum address | SV |
| SR29, 7 | MPUA15 | Protection area maximum address | SV |
| SR30, 7 | MPAT15 | Protection area attribute | SV |

(a) MPM — Memory protection operation mode register

The memory protection mode register is used to define the basic operating state of the memory protection function. The settings for this register are normally specified once on startup and are not changed during program operation.

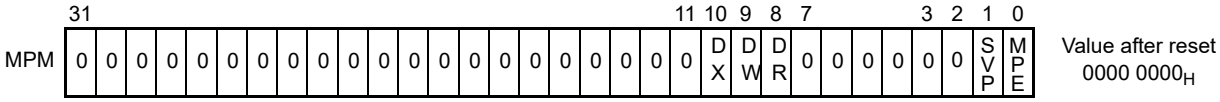



**Table 3.38 MPM Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 11 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 10 | DX | Not supported. It should be set to 0. | R | 0 |
| 9 | DW | Not supported. It should be set to 0. | R | 0 |
| 8 | DR | Not supported. It should be set to 0. | R | 0 |
| 7 to 2 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 1 | SVP | In SV mode. (when PSW.UM = 0), this bit is used to specify whether to restrict access according to the SX, SW, and SR bits of the MPAT register for each protection area.*1<br>0: As usual, implicitly enable all access in SV mode.<br>1: Restrict access according to the SX, SW, and SR bits even in SV mode.*2 | R/W | 0 |
| 0 | MPE | This bit is used to specify whether to enable or disable MPU function.<br>0: Disable<br>1: Enable | R/W | 0 |

Note 1. When the SVP bit is set to 1, access will become restricted after a certain number of clock cycles, even in SV mode. Therefore, specify the protection area before setting the SVP bit to prevent restriction of access by the program itself.

Note 2. If access is restricted in SV mode, execution of MDP exceptions or the MIP exception handling itself might not be possible depending on the settings. Be careful to specify settings so that access to the memory area necessary for the exception handler and exception handling is permitted.

(b) MPRC — MPU region control register

Bits used to perform special memory protection function operations are located in this register.

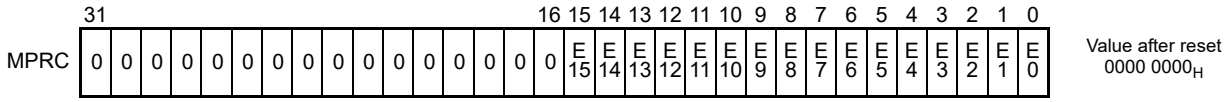



**Table 3.39 MPRC Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 16 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 15 to 0 | E15-E0 | These are the enable bits for each protection area. Bit En is a copy of bit MPATn.E (where n = 15 to 0). | R/W | 0 |

(c) MPBRGN — MPU base region register

This register indicates the minimum usable MPU area number.

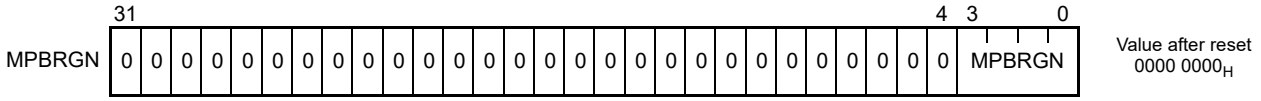



**Table 3.40 MPBRGN Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 4 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 3 to 0 | MPBRGN | These bits indicate the smallest number of an MPU area. These bits are always read as 0 . | R | 0 |

(d) MPTRGN — MPU end region register

This register indicates the maximum usable MPU area number + 1.

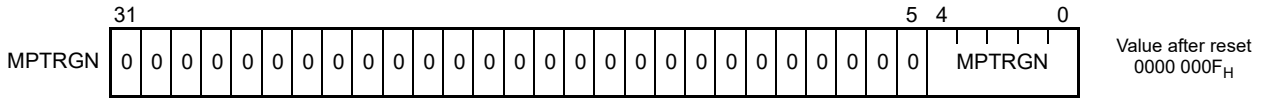



**Table 3.41 MPTRGN Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 5 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 4 to 0 | MPTRGN | These bits always indicate the maximum number of MPU areas that the hardware can support.<br>For P1x-C, number of MPU area is 15. | R | $F_H$ |

(e) MCA — Memory protection setting check address register

This register is used to specify the base address of the area for which a memory protection setting check is to be performed.

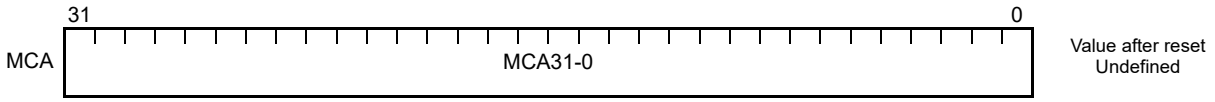



**Table 3.42 MCA Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 0 | MCA31-0 | These bits are used to specify the start address of the memory area subject to a memory protection setting check in bytes. | R/W | Undefined |

(f) MCS — Memory protection setting check size register

This register is used to specify the size of the area for which a memory protection setting check is to be performed.

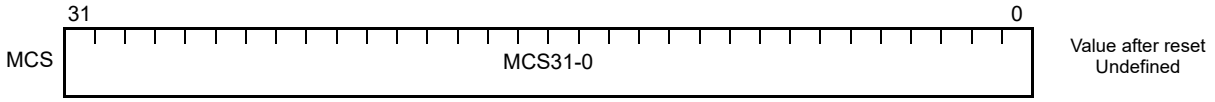



**Table 3.43 MCS Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 0 | MCS31-0 | These bits specify the size of the target area to specify the size of the memory area subject to a memory protection setting check in bytes. These bits specify the size in bytes of a memory area that is subject to a memory protection setting. Checking in areas below the address value in the MCS register is not possible because the specified size is handled as an unsigned integer.<br>Do not set 0000 $0000_H$ in the MCS register. | R/W | 0 |

(g) MCC — Memory protection setting check command register

This command register is used to start a memory protection setting check.

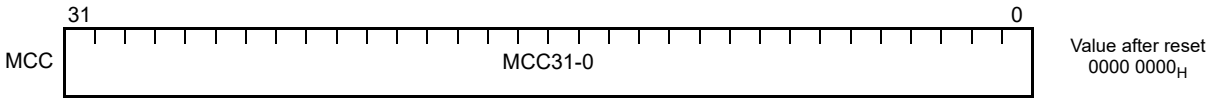



**Table 3.44 MCC Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 0 | MCC31-0 | When any value is written to the MCC register, a memory protection setting check starts. Setting the MCA and MCS registers and then writing to this register leads to storage of the result of checking in the MCR register.<br>Since writing any value to this register starts the check, doing so does not require any extra registers when r0 is used as a source register.<br>The result of checking is reflected in MCR according to any area setting regardless of the setting of the PSW.UM bit.<br>The value read from the MCC register is always 0000 $0000_H$. | R/W | 0 |

(h) MCR — Memory protection setting check result register

This register is used to store the results of a memory protection setting check.

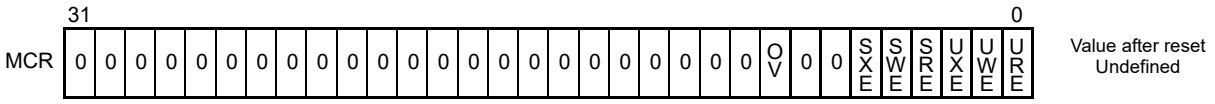



**Table 3.45 MCR Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 9 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 8 | OV | If the specified area includes 0000 $0000_H$ or 7FFF $FFFF_H$, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 7, 6 | — | (Reserved for future expansion. Be sure to clear to 0.) | R/W | 0 |
| 5 | SXE | If the specified area is contained within one protection area and execution is permitted for that area in supervisor mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 4 | SWE | If the specified area is contained within one protection area and writing to that area is permitted in supervisor mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 3 | SRE | If the specified area is contained within one protection area and reading from that area is permitted in supervisor mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 2 | UXE | If the specified area is contained within one protection area and execution is permitted for that area in user mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 1 | UWE | If the specified area is contained within one protection area and writing to that area is permitted in user mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 0 | URE | If the specified area is contained within one protection area and reading from that area is permitted in user mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |

(i) MPLAn — Protection area minimum address register

These registers indicate the minimum address of area n (where n = 0 to 15).

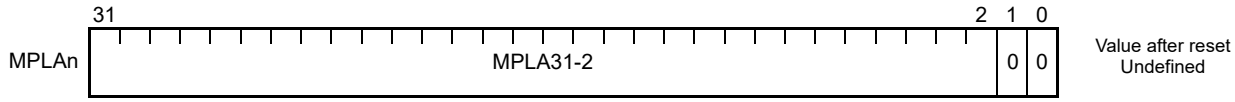



**Table 3.46 MPLAn Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 2 | MPLA31-2 | These bits indicate the minimum address of area n.<br>The MPLA1, 0 bits are used implicitly set to 0. | R/W | Undefined |
| 1, 0 | — | Reserved for future expansion. Be sure to clear these bits to 0. | R | 0 |

(j) MPUAn — Protection area maximum address register

These registers indicate the maximum address of area n (where n = 0 to 15).

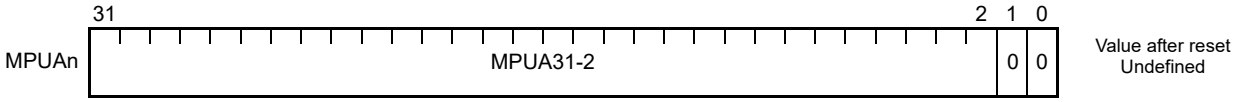



**Table 3.47 MPUAn Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 2 | MPUA31-2 | These bits indicate the maximum address of area n.<br>The MPUA1, 0 bits are used implicitly set to 1. | R/W | Undefined |
| 1, 0 | — | Reserved for future expansion. Be sure to clear these bits to 0. | R | 0 |

(k) MPATn — Protection area attribute register

These registers indicate the attributes of area n (where n = 0 to 15).

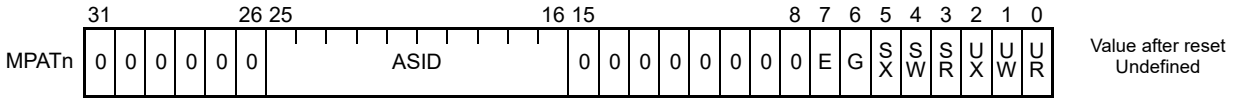



**Table 3.48 MPATn Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 26 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 25 to 16 | ASID | These bits indicate the ASID value to be used as the area match condition. | R/W | Undefined |
| 15 to 8 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 7 | E | This bit indicates whether area n is enabled or disabled. | R/W | 0 |
| 6 | G | This bit is used for the global specification of an MPU area. When this bit is set to 1, the ASID bit has no effect on the area-matching condition. | R/W | Undefined |
| 5 | SX | This bit indicates the execution privilege for supervisor mode.*1 | R/W | Undefined |
| 4 | SW | This bit indicates the write permission for supervisor mode.*1 | R/W | Undefined |
| 3 | SR | This bit indicates the write permission for supervisor mode*1 | R/W | Undefined |
| 2 | UX | This bit indicates the execution privilege for user mode. | R/W | Undefined |
| 1 | UW | This bit indicates the write permission for user mode. | R/W | Undefined |
| 0 | UR | This bit indicates the read permission for user mode. | R/W | Undefined |

Note 1. If access is restricted in SV mode., execution of MDP exceptions or the MIP exception handling itself might not be possible depending on the settings. Be careful to specify settings so that access to the memory area necessary for the exception handler and exception handling is permitted.

### (6) Cache Operation Function registers

The cache control system registers are individually provided for both physical CPU.

**Table 3.49 Cache Operation Function Registers**

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR16, 4 | ICTAGL | Instruction cache tag Lo access | SV |
| SR17, 4 | ICTAGH | Instruction cache tag Hi access | SV |
| SR18, 4 | ICDATL | Instruction cache data Lo access | SV |
| SR19, 4 | ICDATH | Instruction cache data Hi access | SV |
| SR24, 4 | ICCTRL | Instruction cache control | SV |
| SR26, 4 | ICCFG | Instruction cache configuration | SV |
| SR28, 4 | ICERR | Instruction cache error | SV |

#### (a) ICTAGL — Instruction cache tag Lo access register

This register is used for CIST and CILD instructions for the instruction cache. This register holds values to be stored in the tag RAM of the instruction cache by the execution of CIST instructions and values read from the tag RAM of the instruction cache by the execution of CILD instructions.

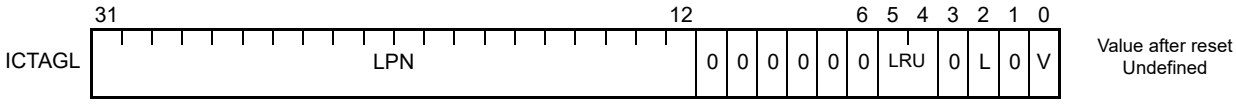



**Table 3.50 ICTAGL Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 12 | LPN | These bits hold the values of 31 to 12 bits of physical page numbers, i.e. the physical page numbers. When writing, always write 0 to bits 31 to 25. | R/W | Undefined |
| 11 to 6 | — | Reserved for future expansion. When writing, always write 0 to these bits. | R | 0 |
| 5, 4 | LRU | These bits indicate the LRU information of the specified cache line. The CIST instruction cannot be used to change the LRU information to desired values. | R/W | Undefined |
| 3 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 2 | L | This bit holds the lock information. | R/W | Undefined |
| 1 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 0 | V | This bit retains whether the specified cache line is enabled or disabled. | R/W | Undefined |

(b) ICTAGH — Instruction cache tag Hi access register

This register is used for a CIST or CILD instruction for the instruction cache. This register stores the value to be stored in the instruction cache tag RAM on CIST execution and the value read from the instruction cache tag RAM on CILD execution.

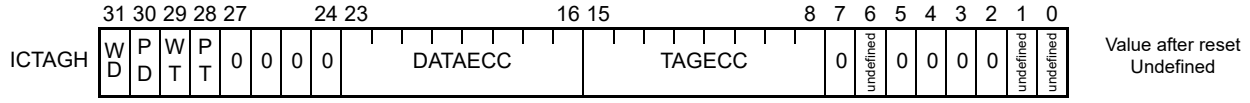



Table 3.51 ICTAGH Register Contents

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 | WD | When this bit is set to 1, executing a CIST instruction updates the cache data RAM. | R/W | Undefined |
| 30 | PD | When this bit is set to 1, executing a CIST instruction writes the value in the DATAECC field of this register to the ECC/parity field of the data RAM. | R/W | Undefined |
| 29 | WT | This bit updates the cache tag RAM when this bit is set to 1 before CIST execution. | R/W | Undefined |
| 28 | PT | This bit writes the value in TAGECC field to ECC for tag RAM when this bit is set to 1 before CIST execution. | R/W | Undefined |
| 27 to 24 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 23 to 16 | DATAECC | These bits retain ECC/parity of the data RAM. | R/W | Undefined |
| 15 to 8 | TAGECC | These bits retain ECC of the tag RAM. Be sure to clear bit 15 to 0. | R/W | Undefined |
| 7 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 6 | — | (Reserved for future expansion. Be sure to set to 0.) | R | Undefined |
| 5 to 2 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 1, 0 | — | (Reserved for future expansion. Be sure to set to 0.) | R | Undefined |

(c) ICDATL — Instruction cache data Lo access register

This register is used for a CIST or CILD instruction for the instruction cache. This register stores the value to be stored in the instruction cache tag RAM on CIST execution and the value read from the instruction cache tag RAM on CILD execution.

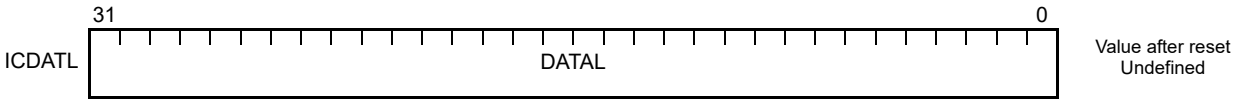



Table 3.52 ICDATL Register Contents

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 0 | DATAL | These bits retain the values of bits 31 to 0 or of bit 95 to 64 from the instruction data of the block within the specified cache line. The offset of the index specifies the target range of bit numbers.<br>Index offset = 0000: bits 31 to 0<br>Index offset = 1000: bits 95 to 64 | R/W | Undefined |

### (d) ICDATH — Instruction cache data Hi access register

This register is used for a CIST or CILD instruction for the instruction cache. This register stores the value to be stored in the instruction cache tag RAM on CIST execution and the value read from the instruction cache tag RAM on CILD execution

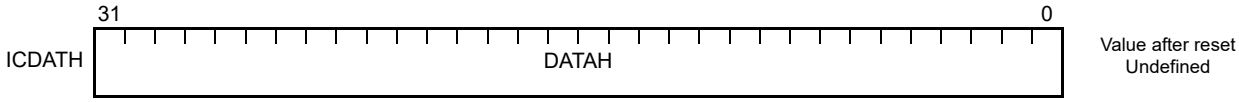



**Table 3.53 ICDATH Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 0 | DATAH | These bits retain the values of bit 63 to 32 or those of bit 127 to 96 from the instruction data of the block within the specified cache line. The index offset specifies the bit number to be retained.<br>Index offset = 0000: bit 63 to 32<br>Index offset = 1000: bit 127 to 96 | R/W | Undefined |

### (e) ICCTRL — Instruction cache control register

This register controls the instruction cache.

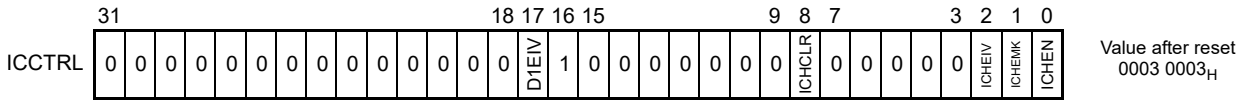



**Table 3.54 ICCTRL Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 18 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 17 | D1EIV | This bit selects the operation in response to 1-bit errors in the data RAM.<br>0: The error is corrected and then processing continues, but the address of the entry that had an error is retained.<br>1: The error is not corrected, the entry is cleared, and fetching is repeated.<br>This bit is read as the previous value until the setting is actually reflected in the instruction cache. | R/W | 1 |
| 16 | — | (Reserved for future expansion. Be sure to set to 1.) | R | 1 |
| 15 to 9 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 8 | ICHCLR | Setting this bit to 1 selects clearing of the whole instruction cache in a single operation. After this bit has been set to 1, it will be read as 1 until clearing is completed. The bit is cleared to 0 once clearing of the cache is completed. | R/W | 0 |
| 7 to 3 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 2 | ICHEIV | Setting this bit to 1 allows the instruction cache to be automatically disabled (ICHEN bit to be cleared to 0) when a cache error occurs. | R/W | 0 |
| 1 | ICHEMK | Setting this bit to 1 selects masking of cache error exception notifications for the CPU when cache errors are encountered. | R/W | 1 |
| 0 | ICHEN | This bit disables or enables the instruction cache.<br>0: Instruction cache is disabled.<br>1: Instruction cache is enabled.<br>This bit is read as the previous value until the setting is actually reflected in the instruction cache. | R/W | 1 |

(f) ICCFG — Instruction cache configuration register

This register shows the configuration of the instruction cache.

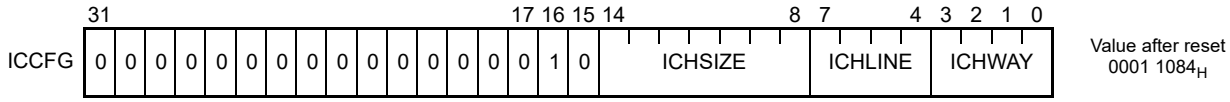



**Table 3.55 ICCFG Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 17 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 16 | — | (Reserved for future expansion. Be sure to set to 1.) | R | 1 |
| 15 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 14 to 8 | ICHSIZE | These bits indicate the capacity (in K bytes) of the instruction cache.<br>001 0000: 16 K byte | R | $10_H$ |
| 7 to 4 | ICHLINE | These bits indicate the number of lines per 1 way in the instruction cache.<br>1000: 256 lines | R | $8_H$ |
| 3 to 0 | ICHWAY | These bits indicate the number of ways of the instruction cache.<br>0100: 4 ways | R | $4_H$ |

### (g) ICERR — Instruction cache error register

This register stores cache error data of the instruction cache.

Once the ICHERR bit is set to 1, subsequent cache error data is not stored in this register until the ICHERR bit is cleared to 0.

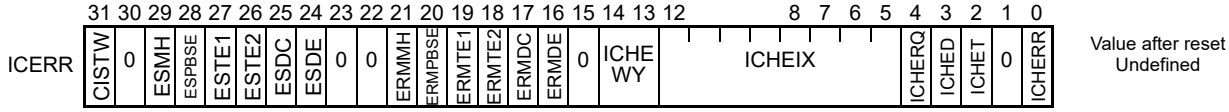



**Table 3.56 ICERR Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 | CISTW | This bit is set to indicate that the destination way specified for a CISTI instruction was in error. Although the entry information is overwritten so that writing is completed, the V bit will be cleared the next time the cache line is read (i.e. reading will be judged to have missed the cache). However, setting of this bit is not accompanied by an exception for the CPU. | R/W | 0 |
| 30 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 29 | ESMH | Error status: multi-hit | R/W | Undefined |
| 28 | ESPBSE | Error status: way error | R/W | Undefined |
| 27 | ESTE1 | Error status: 1-bit error in the tag RAM | R/W | Undefined |
| 26 | ESTE2 | Error status: 2-bit error in the tag RAM | R/W | Undefined |
| 25 | ESDC | Error status: 1-bit correction in the data RAM | R/W | Undefined |
| 24 | ESDE | Error status: 2-bit correction in the data RAM | R/W | Undefined |
| 23, 22 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 21 | ERMMH | Error exception notification mask: multi-hit | R/W | 0 |
| 20 | ERMPBSE | Error exception notification mask: way error | R/W | 0 |
| 19 | ERMTE1 | Error exception notification mask: 1-bit error in the tag RAM | R/W | 0 |
| 18 | ERMTE2 | Error exception notification mask: 2-bit error in the tag RAM | R/W | 0 |
| 17 | ERMDC | Error exception notification mask: 1-bit correction in the data RAM | R/W | 0 |
| 16 | ERMDE | Error exception notification mask: 2-bit correction in the data RAM | R/W | 0 |
| 15 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 14, 13 | ICHEWY | These bits retain a way number where a cache error occurs. | R/W | Undefined |
| 12 to 5 | ICHEIX | These bits retain a cache index where a cache error occurs. | R/W | Undefined |
| 4 | ICHERQ | Setting of this bit to 1 indicates that the CPU is being notified of a cache error exception. If cache error exceptions are masked, however, the CPU is not notified of an exception even when this bit is set to 1. | R/W | 0 |
| 3 | ICHED | This bit indicates that an error occurs in the data RAM. | R/W | 0 |
| 2 | ICHET | This bit indicates that an error occurs in the tag RAM. | R/W | 0 |
| 1 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 0 | ICHERR | This bit is set to 1 when a cache error occurs. | R/W | 0 |

### (7) Data Buffer Operation Function Registers

The data buffer control system registers can be individually provided of the physical CPU.

**Table 3.57 Data Buffer Operation Function Registers**

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|---|---|---|---|
| SR24, 13 | CDBCR | Data buffer control register | SV |

#### (a) CDBCR — Data Buffer Control Register

This register controls the data buffer.

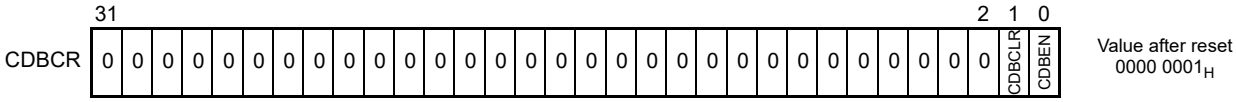



**Table 3.58 CDBCR Register Contents**

| Bit | Name | Description | R/W | Value after reset |
|---|---|---|---|---|
| 31 to 2 | — | (Reserved for future expansion. Be sure to clear to 0.) | R | 0 |
| 1 | CDBCLR | When this bit is set to 1, the data buffer is cleared at a time. This bit is always read as 0. | W | 0 |
| 0 | CDBEN | This bit disables or enables the data buffer.<br>0: Data buffer is disabled.<br>1: Data buffer is enabled. | R/W | 1 |

## 3.2.2 Instruction Cache and Data Buffer

### 3.2.2.1 Features

A 16-kbyte and 4-way set-associative Instruction Cache is mounted between the CPU1, 2 and the Code Flash. The Instruction Cache and the Code Flash are connected to each other via a 128-bit dedicated bus to minimize penalties caused by a cache miss-hit. Also a Data Buffer is mounted between the CPU1, 2 and the Code Flash to achieve high-speed data access. The 32-MB area from $0000\ 0000_H$ to $01FF\ FFFF_H$ in the address space is intended for the Instruction Cache and Data Buffer.

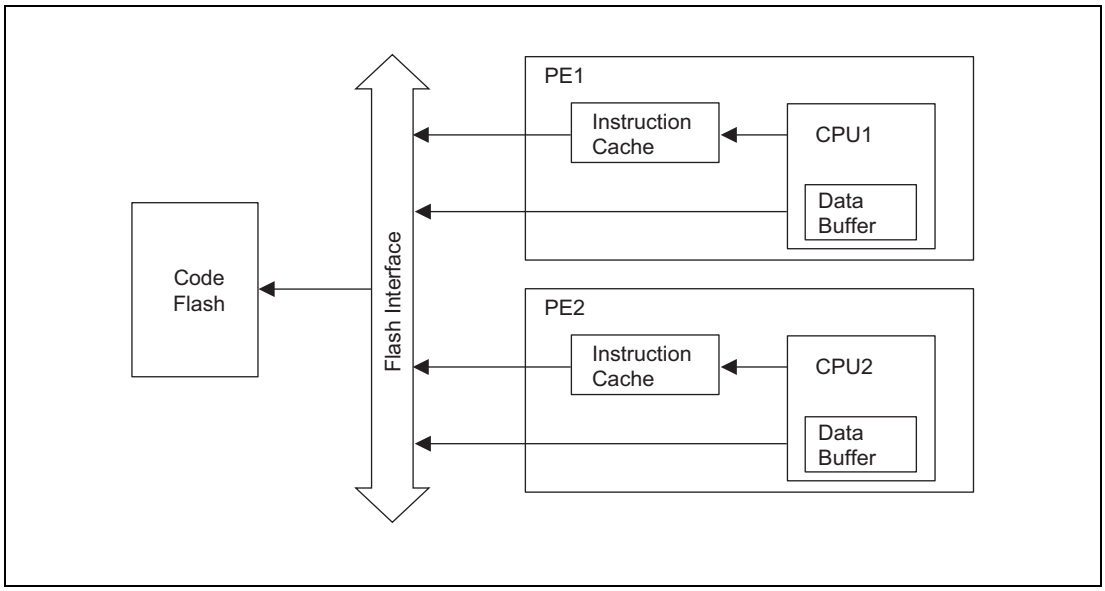



**Figure 3.2 Instruction Cache and Data Buffer**

### 3.2.2.2 Instruction Cache Function

The 16-kbyte and 4-way set-associative Cache includes four Ways consisting of 256-entry blocks of four words per line, amounting to 16 kbytes capacity in total. The Ways are divided into two groups, Way Group 0 consisting of Way0 and Way1 and Way Group 1 consisting of Way2 and Way3. The Way Group can be selected and used by decoding of the address information of the access destination. If a cache error occurs, each line is refilled by a replace algorithm using LRU. CPU instruction fetches from the Code Flash are performed via the instruction cache.

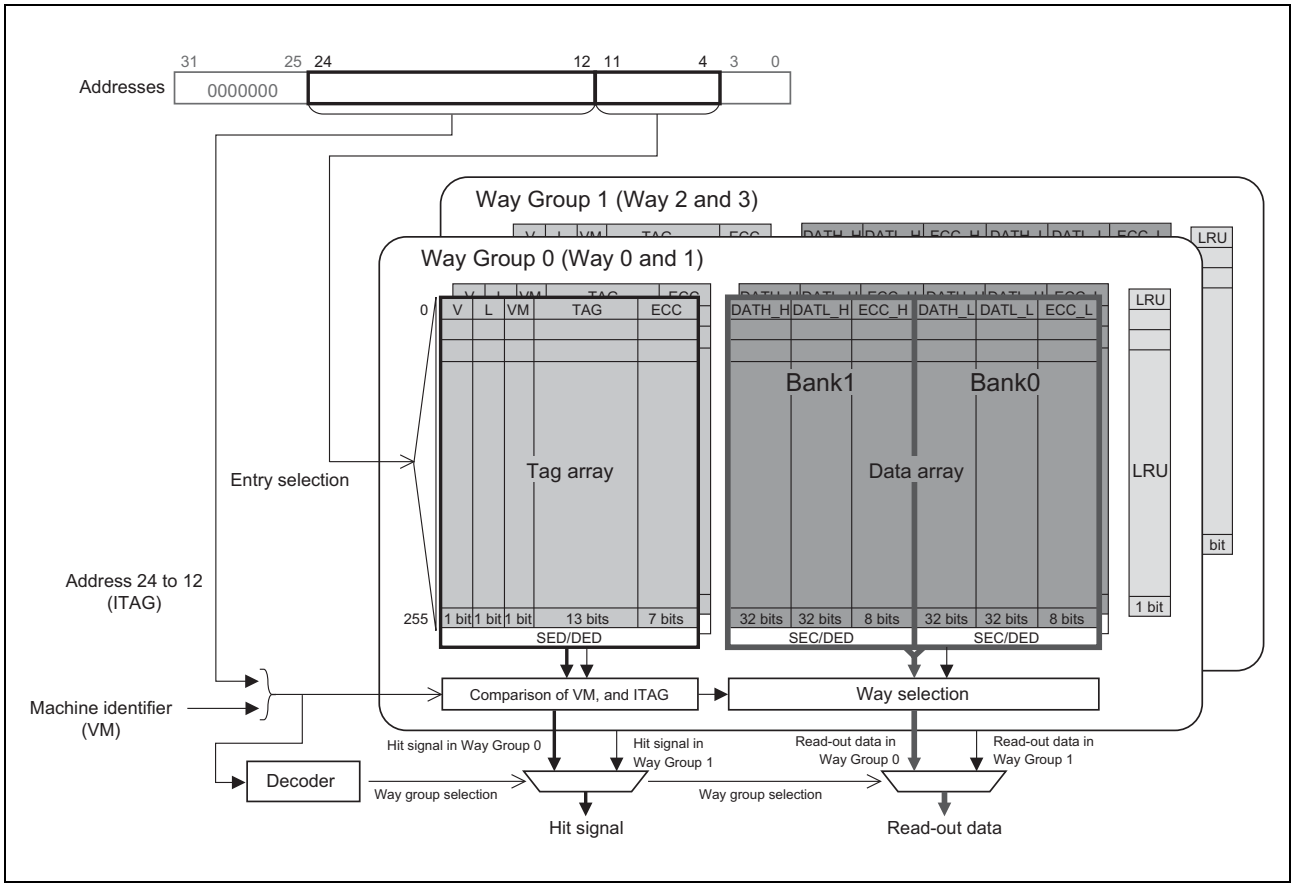



**Figure 3.3 Instruction Cache Configuration**

**Tag Array**

| | |
|---|---|
| V bit | This bit indicates whether valid data is stored in the cache line. Setting of this bit to 1 makes the cache line data valid. The V bit is initialized to 0 by reset. |
| L bit | This bit indicates whether a cache line is locked or not. Setting of this bit to 1 locks the cache line and it can not be replaced with new data. The L bit is valid only when the V bit is 1, and it is not initialized by reset. |
| VM bit | VM bit is always set into 0 because virtual machine mode is not supported.<br>The VM bit is not initialized by reset. |
| TAG | Among 32 bits in the operable addresses of the data line to be cached, bits 24 to 12 are stored in this bit. The TAG bit is not initialized by reset. |
| ECC | The ECC of the tag array is stored in this bit. The ECC bit is not initialized by reset. |

**Data Array**

| | |
|---|---|
| DATH_H, DATH_L, DATL_H, DATL_L | The 128-bit cache line data is stored per 32 bits as follows: the bits [127:96], [95:64], [63:32], [31:0] in the DATH_H, DATL_H, DATH_L, and DATL_L, respectively. In the CIST or CILD operation in response to a cache instruction, the ICDATH register is used for DATH_H and DATH_L and ICDATL is used for DATL_H and DATL_L. |
| ECC_H, ECC_L | The ECC of the data in bits [127:64] and [63:0] are stored in ECC_H and ECC_L, respectively. |

**LRU**

| | |
|---|---|
| LRU | The LRU information in the same Way Group is stored in this data array. The LRU is initialized by reset. |

**CAUTION**

**When writing test data to the tag array of the instruction cache in the CIST instruction and then fetching the relevant line by the instruction, write the tag information in units of WAY group. For example, when writing tag information to a line on the WAY0 side of WAY group 0, also write the tag information of the same line on the WAY1 side and then perform instruction fetch.**

- **When writing data to WAY group 0 (WAY0, WAY1), write a value that makes the Exclusive OR of the ICTAGL.LPN bit bit 0.**
- **When writing data to WAY group 1 (WAY2, WAY3), write a value that makes the Exclusive OR of the ICTAGL.LPN bit bit 1.**

**When a value other than those above is written to the tag array and instruction fetch is performed, a WAY error occurs and the ICERR.ESPBSE bit is set to 1. When the same tag information is written to the same line of two WAYs in a WAY group and instruction fetch is performed, a multi-hit error occurs and the ICERR.ESMH bit is set to 1.**

### 3.2.2.3 Data Buffer Function

The four-line buffer with 128 bits per line is mounted as a data buffer. The data of 128 bits per line read from the Code Flash is stored in the data buffer. The data is read out from the data buffer after the next access to the same address, so the Code Flash is not accessed again.

### 3.2.3 Inter-Processor Interrupts

Registers (IPIR_CH#) for interrupt communication between PEs are provided for two channels.

IPIR_CH1 to IPIR_CH2 are assigned to CH1 to CH2 of user interrupt (EIINT). An interrupt for specific PEs (including own PE) can be requested by manipulating bits corresponding to respective PEs.

See **Section 6**, Exception/Interrupt Functions.

### 3.2.4 Reliability Functions

#### 3.2.4.1 PE Guard Function (PEG)

**(1) Overview of the PEG Function**

The PEG is a constituent of the slave guard system to prevent unauthorized access to the resources in the PE from the external master. This function protects access to the local RAM in the PE. In the initial state after a reset, all access by masters other than the PE itself is disabled except read access from ICUMC. Setting the registers listed in (3), List of PEG Protection Setting Registers, enables access by masters other than the PE itself.

(1) Detecting PE guard violation

If the external master outside the PE makes an unauthorized access to the resource area in the PE for which PE guard is set, the access is detected as a PE guard violation. PE Guard informs it to ECM and reports the details about the access in registers.

(2) Blocking unauthorized accesses

When a PE guard violation is detected, unauthorized accesses to the internal resources of the PE are blocked to prevent the contents of PE resources from being modified illegally.

(3) Notifying occurrence of violation

An error response to an unauthorized access is sent to the request source of external master (except H-Bus master peripherals).

When the DMAC or DTS makes an unauthorized access, meanwhile, a DMA transfer error is detected.

**(2) Protection Made by SPID**

- Setting PEG Protection
  - Up to eight areas can be set depending on the Local RAM address of the own PE.
  - The area range is specified by the base address and the mask bit (4 kbytes to 4 Gbytes).
  - “Read enable” and “write enable” can be set for each area.
  - “Enable” or “disable” can be selected on each system protection identifier (SPID) for each area.
- Access permission by the system protection identifier (SPID) (see **Figure 3.4**)

1. When the Local RAM area is to be accessed, go to step 2.
   Otherwise, return an error response.
2. When any of enabled area 0 to area 7 is to be accessed, go to step 3.
   Otherwise, return an error response.
3. Are all the conditions below for the relevant area met?

– The system protection identifier (SPID) is enabled.
– Required operations (read/write) are enabled.
Otherwise, return an error response.

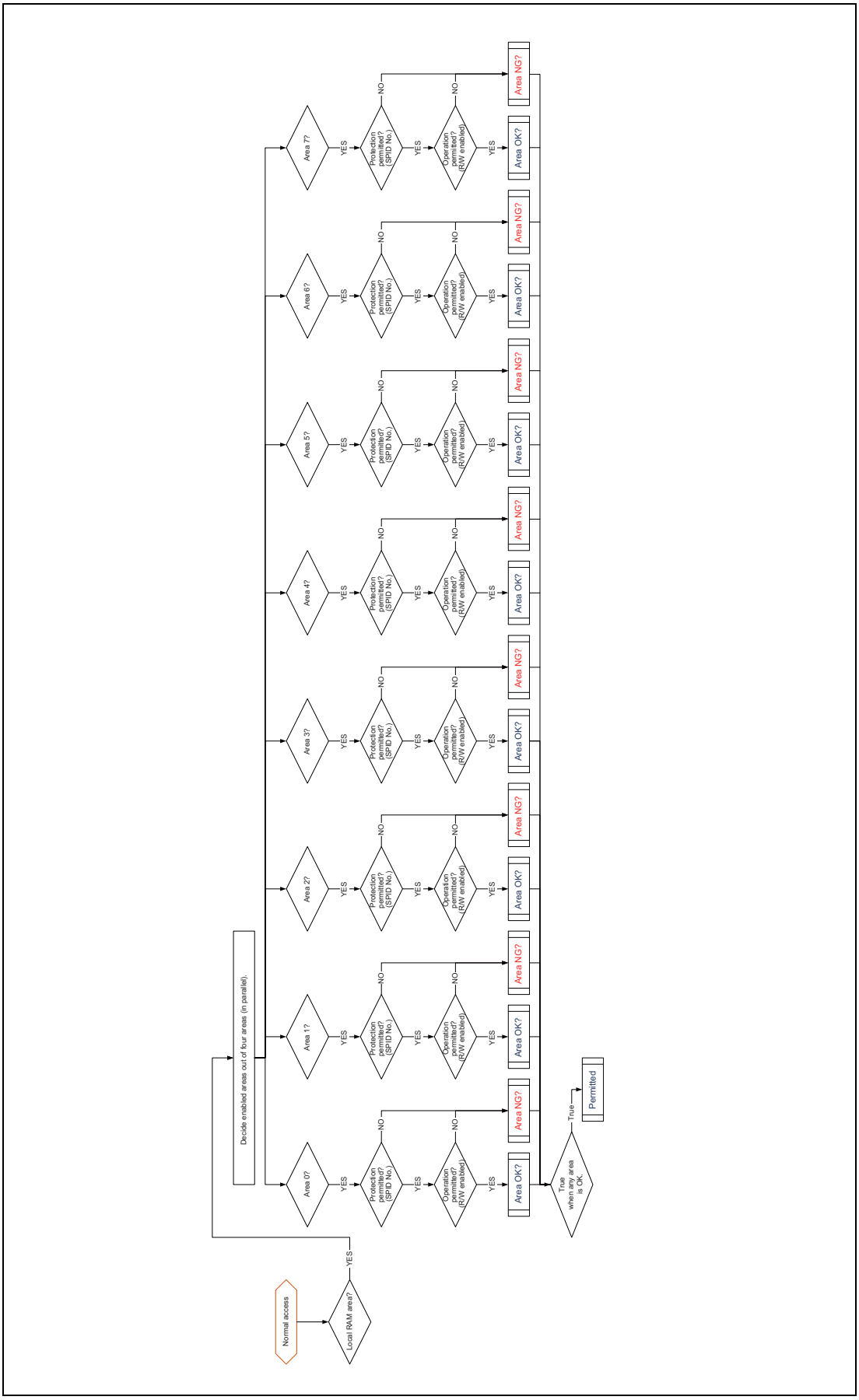



**Figure 3.4 Access Permission by the System Protection Identifier (SPID)**

### (3) List of PEG Protection Setting Registers

Make necessary settings for the following registers to protect PE resources from unauthorized accesses by the external master.

- Accesses to the Local RAM in the PE are permitted as detection targets.

**Table 3.59 PEG registers (Base Address: FFFE E600$_H$)**

| Address Offset | Size (Byte) | Register Name | Abbreviation | Right | R/W | Operable Bit 1 | 8 | 16 | 32 | Value after reset |
|---|---|---|---|---|---|---|---|---|---|---|
| +080$_H$ | 4 | PE guard area 0 mask setting register | PEGG0MK | — | R/W | — | √ | √ | √ | 003F F000$_H$ |
| +084$_H$ | 4 | PE guard area 0 base setting register | PEGG0BA | — | R/W | — | √ | √ | √ | FE80 0003$_H$ |
| +088$_H$ | 4 | PE guard area 0 enable setting register | PEGG0SP | — | R/W | — | √ | √ | √ | 0000 0003$_H$ |
| +090$_H$ | 4 | PE guard area 1 mask setting register | PEGG1MK | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +094$_H$ | 4 | PE guard area 1 base setting register | PEGG1BA | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +098$_H$ | 4 | PE guard area 1 enable setting register | PEGG1SP | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0A0$_H$ | 4 | PE guard area 2 mask setting register | PEGG2MK | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0A4$_H$ | 4 | PE guard area 2 base setting register | PEGG2BA | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0A8$_H$ | 4 | PE guard area 2 enable setting register | PEGG2SP | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0B0$_H$ | 4 | PE guard area 3 mask setting register | PEGG3MK | — | R/W | — | √ | √ | √ | 0000 0000v |
| +0B4$_H$ | 4 | PE guard area 3 base setting register | PEGG3BA | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0B8$_H$ | 4 | PE guard area 3 enable setting register | PEGG3SP | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0C0$_H$ | 4 | PE guard area 4 mask setting register | PEGG4MK | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0C4$_H$ | 4 | PE guard area 4 base setting register | PEGG4BA | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0C8$_H$ | 4 | PE guard area 4 enable setting register | PEGG4SP | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0D0$_H$ | 4 | PE guard area 5 mask setting register | PEGG5MK | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0D4$_H$ | 4 | PE guard area 5 base setting register | PEGG5BA | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0D8$_H$ | 4 | PE guard area 5 enable setting register | PEGG5SP | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0E0$_H$ | 4 | PE guard area 6 mask setting register | PEGG6MK | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0E4$_H$ | 4 | PE guard area 6 base setting register | PEGG6BA | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0E8$_H$ | 4 | PE guard area 6 enable setting register | PEGG6SP | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0F0$_H$ | 4 | PE guard area 7 mask setting register | PEGG7MK | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0F4$_H$ | 4 | PE guard area 7 base setting register | PEGG7BA | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |
| +0F8$_H$ | 4 | PE guard area 7 enable setting register | PEGG7SP | — | R/W | — | √ | √ | √ | 0000 0000$_H$ |

**Table 3.60 PE guard Error Status Registers (Base Address of PE1: FFC4 A200$_H$, PE2: FFC4 A300$_H$)**

| Address Offset | Register Name | Register Symbol | PBG | R/W | Operable Bit 1 | 8 | 16 | 32 | Value after reset |
|---|---|---|---|---|---|---|---|---|---|
| +00h | PE guard Error Status Control Register | PGERRSTATCTL_PEx | *1 | R/W | — | — | — | √ | 0000 0000$_H$ |
| +04h | PE guard Error Status Register | PGERRSTAT_PEx | *1 | R | — | — | — | √ | 0000 0000$_H$ |
| +08h | PE guard Error Information Register | PGERRiNFO_PEx | *1 | R | — | — | — | √ | 0000 0000$_H$ |

Note 1. In case of
x=1 APBGRD_PFSS1.SP1
x=2 APBGRD_PFSS1.SP2

x = 1, 2

### (4) Register Set

#### (a) PEGGnMK — PE Guard Area n Mask Setting Register (n = 0 to 7)

The PEGGnMK register defines which bits of PEGGnBA are compared with the access address. If bit MASKm is cleared, bit BASEm is compared with bit m of the access address.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GnMASK | | | | | | | | | | | | | | | |
| Value after reset *1 | | | | | | | | | | | | | | | | |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GnMASK | | | | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset *1 | | | | | | | | | | | | | | | | |
| | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. See **Table 3.59, PEG registers (Base Address: FFFE E600$_H$)**.

**Table 3.61 GnMask register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 12 | GnMASK | 0: Target address bits are compared when determining the PE guard area.<br>1: Target address bits are not compared when determining the PE guard area. |
| 11 to 0 | Reserved | These bits are always read as 0. The write value should always be 0. |

(b) PEGGnBA — PE Guard Area n Base Setting Register (n = 0 to 7)

In combination with the PEGGnMK register, this register specifies a range or ranges within PE guard protection area n. Setting the GnEN bit to 1 brings the address enable conditions specified by this register and the PEGGnMK register into effect.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GnBASE | | | | | | | | | | | | | | | |
| Value after reset *1 | | | | | | | | | | | | | | | | |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GnBASE | | | | — | — | — | — | — | — | — | GnLOCK | — | GnWR | GnRD | GnEN |
| Value after reset *1 | | | | | | | | | | | | | | | | |
| | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | W | R | R/W | R/W | R/W |

Note 1. See **Table 3.59, PEG registers (Base Address: FFFE E600$_H$)**.

**Table 3.62 PEGGnBA register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 12 | GnBASE | Base address that specifies PE guard protection area n |
| 11 to 5 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 4 | GnLOCK | Enables write protect to the registers (PEGGnBA, PEGGnMK, PEGGnSP) of PEGuard control. This bit can be written once after reset.<br>0: Registers are not protected.<br>1: Registers are protected. |
| 3 | Reserved | This bit is always read as 0. The write value should always be 0. |
| 2 | GnWR | Enables write access to PE guard protection area n.<br>0: Write access is disabled.<br>1: Write access is enabled. |
| 1 | GnRD | Enables read access to PE guard protection area n.<br>0: Read access is disabled.<br>1: Read access is enabled. |
| 0 | GnEN | PE guard protection area n enable<br>0: Settings for access enable conditions are disabled<br>1: Settings for access enable conditions are enabled |

(c) PEGGnSP — PE Guard Area n enable Setting Register (n = 0 to 7)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GnSPm | | | | | | | | | | | | | | | |
| Value after reset *1 | | | | | | | | | | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GnSPm | | | | | | | | | | | | | | | |
| Value after reset *1 | | | | | | | | | | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note 1. See **Table 3.59, PEG registers (Base Address: FFFE E600$_H$)**.

**Table 3.63 PEGGnSP register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | GnSPm | Each set bit m of this register allows the access to the region n by the SPID m. (m = 0 to 31, shows bit position)<br>0: Access with SPID is not allowed.<br>1: Access with SPID is allowed. |

(d) PGERRSTATCTL_PEx — PE Guard Error Status Control Register

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLRO | CLRE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 3.64 PGERRSTATCTL_PEx register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | CLRO | Clear the OVF bit of PGERRSTAT_PEx by writing this bit to “1”.<br>0: Clear is completed.<br>1: Clear is on execution. |
| 0 | CLRE | Clear the ERR bit of PGERRSTAT_PEx by writing this bit to “1”.<br>0: Clear is completed.<br>1: Clear is on execution. |

(e) PGERRSTAT_PEx — PE Guard Error Status Register

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF | ERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 3.65 PGERRSTAT_PEx register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset always is read. |
| 1 | OVF | 0: No or one access violation has occurred<br>1: More than one access violation has occurred |
| 0 | ERR | 0: No access violations have occurred<br>1: At least one access violation has occurred |

(f) PGERRINFO_PEx — PE Guard Error Information Register

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | PEID | | | SPID | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 3.66 PGERRINFO_PEx register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset always is read. |
| 7 to 5 | PEID | PEID of the bus master that initiated the access |
| 4 to 0 | SPID | SPID of the bus master that initiated the access |

### 3.2.4.2 PE's Internal Peripherals Protection Function (IPG)

#### (1) Overview of the IPG Function

The IPG is a system to prevent unauthorized accesses to peripherals from the CPU core equipped with the IPG. The IPG achieves the following functions. The IPG covers accesses to resources except the Code Flash and the local RAM.

(1) **Detecting violation of peripherals protection**
If the CPU makes an unauthorized access to an area (peripherals) for which peripherals protection is set, the access is detected as "violation of peripherals protection."

(2) **Storing unauthorized-access information**
When a violation of peripherals protection is detected, the unauthorized-access information is stored in the IPG's internal register.

(3) **Blocking unauthorized accesses**
When a violation of peripherals protection is detected, unauthorized accesses to peripherals are blocked to prevent contents of peripherals from being modified illegally.

(4) **Notifying violation**
When a violation of peripherals protection is detected, a request for generating an exception is made to ask the CPU to stop the processing.

**NOTE**

The G3M makes a request for generating a SYSERR (asynchronous) exception via SysErrGen.

(5) **Invalidating subsequent accesses**
When a violation of peripherals protection is detected, subsequent accesses (regardless of authorized or unauthorized accesses) are blocked until instructions from the CPU are received.

**NOTE**

Even if a request for generating an exception is immediately sent to the CPU in step (4) above, a subsequent access issued (before receiving a request from the IPG) by the CPU that does not know an occurrence of violation may illegally modify contents of peripherals. (Accesses after a violation has occurred result in unauthorized accesses.)

#### (2) IPG Function

(1) This function invalidates accesses according to their attributes (including address, transfer type, and access right).

(2) After an access right violation is detected until the error flag (described later) is cleared by writing by the software, subsequent accesses are invalidated. However, invalidation is applied only to accesses from the CPU and is not applied to accesses from outside the CPU core. Invalidation is performed independently of addresses.

(3) When a request for accessing different peripherals simultaneously is made due to misalignment or double-word access, the access is executed when all such accesses are enabled.

### (3) IPG Protection Setting Registers for Illegal Users

To protect peripherals from unauthorized accesses by programs in user mode, necessary settings are required for the registers listed below.

- Accesses to respective machines in user mode are to be detected.
- This register set is intended for IPG settings related to user mode and reading the IPG settings in own machine.

**Table 3.67 IPG registers (Base Address: FFFE E000$_H$)**

| Address Offset | Size (Byte) | Register Name | Abbreviation | Right*1 | R/W | Operable Bit | | | | Value after reset |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 1 | 8 | 16 | 32 | |
| +002$_H$ | 2 | Peripherals protection violation access information register | IPGECRUM | SV | R/W | — | — | √ | — | Undefined |
| +008$_H$ | 4 | Peripherals protection violation access address register | IPGADRUM | SV | R/W | — | — | — | √ | Undefined |
| +00D$_H$ | 1 | Peripherals protection enable register | IPGENUM | SV | R/W | √ | √ | — | — | 00$_H$ |
| +020$_H$ | 1 | Peripherals protection setting register 0 | IPGPMTUM0 | SV | R/W | √ | √ | — | — | 00$_H$ |
| +021$_H$ | 1 | Peripherals protection setting register 1 | IPGPMTUM1 | SV | R/W | √ | √ | — | — | 00$_H$ |
| +022$_H$ | 1 | Peripherals protection setting register 2 | IPGPMTUM2 | SV | R/W | √ | √ | — | — | 00$_H$ |
| +023$_H$ | 1 | Peripherals protection setting register 3 | IPGPMTUM3 | SV | R/W | √ | √ | — | — | 00$_H$ |
| +024$_H$ | 1 | Peripherals protection setting register 4 | IPGPMTUM4 | SV | R/W | √ | √ | — | — | 00$_H$ |

Note 1. Registers for which “SV” is described are accessible by accesses with SV right (UM = 0).

### (4) Register Set

(a) IPGECRUM — Peripherals Protection Violation Access Information Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DS | | | | EX | WR | RD | VD |
| Value after reset | 0 | 0 | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

×: Undefined

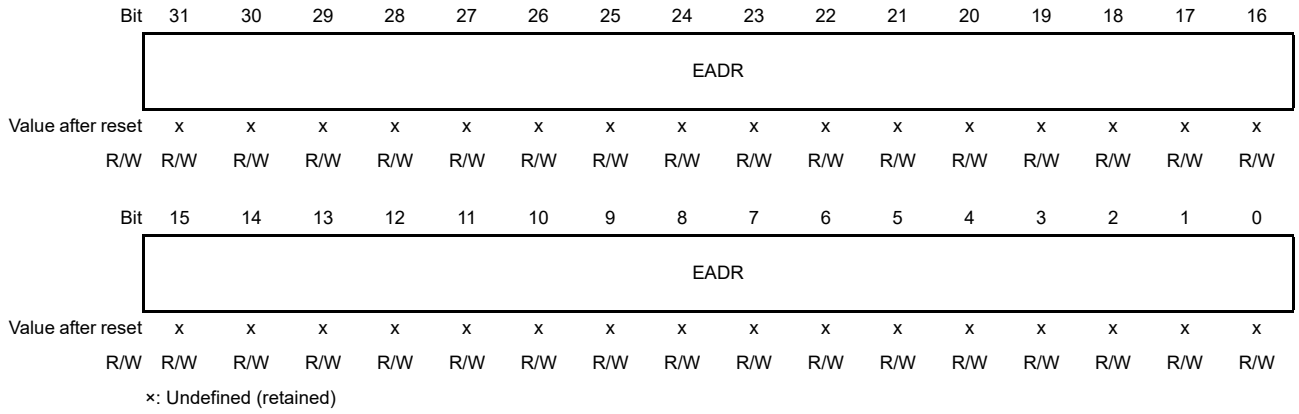
**Table 3.68 Register Contents of IPGECRUM**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15, 14 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 13 to 8 | Reserved | The read value is undefined. The write value should always be 0. |
| 7 to 4 | DS | These bits store the data size of access that made a violation.<br>1000: Double-word (8 bytes)<br>0100: Word (4 bytes)<br>0010: Half-word (2 bytes)<br>0001: Byte Other than above: RFU |
| 3 | EX | This bit is set to 1 when a violation occurred in an instruction fetch read access. In other cases, this bit is cleared to 0. |
| 2 | WR | This bit is set to 1 when a violation occurred in a write access or bit operation or CAXI. In other cases, this bit is cleared to 0. |
| 1 | RD | This bit is set to 1 when a violation occurred in a read access or bit operation or CAXI. In other cases, this bit is cleared to 0. |
| 0 | VD | This bit is set to 1 when a violation of peripherals protection is detected by a program with the relevant right. If another violation of peripheral protection is detected, data of this IPGECRUM register and the IPGADRUM register is updated. |

**NOTE**

When the IRE bit value of the IPGENUM register (described later) is 0 and violation of peripherals protection by a program operating in user mode is an instruction fetch read access, no bit of this register is updated.

(b) IPGADRUM — Peripherals Protection Violation Access Address Register




×: Undefined (retained)

**Table 3.69 Register Contents of IPGADRUM**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | EADR | These bits store the address of the access in which a violation occurred. |

**NOTE**

When the IRE bit value of the IPGENUM register (described later) is 0 and violation of peripherals protection by a program operating in user mode is an instruction fetch read access, no bit of this register is updated.

(c) IPGENUM — Peripheral Device Protection Enable Register

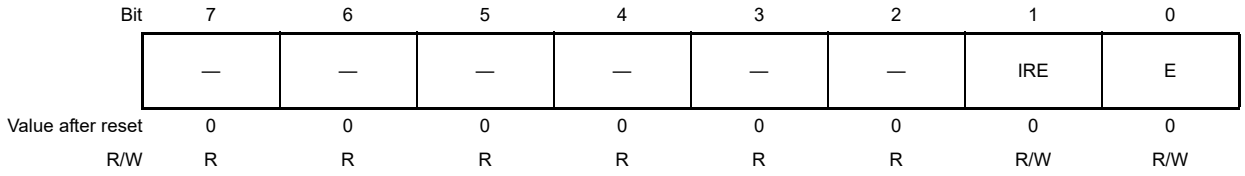



**Table 3.70 Register Contents of IPGENUM**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | IRE | This bit sets whether to store the access information in the peripheral device protection violation access address register and the peripherals protection violation access information register when a violation of peripherals protection occurred in an instruction fetch access.<br>0: Instruction fetch access information is not stored. (Value after reset)<br>1: Instruction fetch access information is stored.<br>**CAUTION**<br>If you do not want to detect speculative instruction fetches (no instruction is executed in some cases), clear this bit to 0. |
| 0 | E | This bit enables or disables the peripherals protection function against accesses by the relevant access right.<br>0: The peripherals protection function is disabled. (Value after reset)<br>1: The peripherals protection function is enabled. |

(d) IPGPMTUM0 — Peripherals Protection Setting Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | X1 | W1 | R1 | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R | R | R | R |

**Table 3.71 Register Contents of IPGPMTUM0**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 6 | X1 | This bit sets whether to enable instruction fetch read access to Peripheral Groups.<br>0: Instruction fetch read access to Peripheral Groups is treated as violation. (Value after reset)<br>1: Instruction fetch read access to Peripheral Groups is not restricted. |
| 5 | W1 | This bit sets whether to enable write access to Peripheral Groups.<br>0: Write access to Peripheral Groups is treated as violation. (Value after reset)<br>1: Write access to Peripheral Groups is not restricted. |
| 4 | R1 | This bit sets whether to enable read access to Peripheral Groups.<br>0: Read access to Peripheral Groups is treated as violation. (Value after reset)<br>1: Read access to Peripheral Groups is not restricted. |
| 3 to 0 | Reserved | Reserved. These bits are always read as 0. The write value should always be 0. |

(e) IPGPMTUM1 — Peripherals Protection Setting Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | X1 | — | — | — | X0 | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R | R | R/W | R | R |

**Table 3.72 Register Contents of IPGPMTUM1**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 6 | X1 | This bit sets whether to enable instruction fetch read access to GRAM Bank#1.<br>0: Instruction fetch read access to GRAM Bank#1 is treated as violation. (Value after reset)<br>1: Instruction fetch read access to GRAM Bank#1 is not restricted. |
| 5 to 3 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 2 | X0 | This bit sets whether to enable instruction fetch read access to GRAM Bank#0.<br>0: Instruction fetch read access to GRAM Bank#0 is treated as violation. (Value after reset)<br>1: Instruction fetch read access to GRAM Bank#0 is not restricted. |
| 1, 0 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |

(f) IPGPMTUM2 — Peripherals Protection Setting Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | W1 | R1 | — | — | W0 | R0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R | R/W | R/W |

**Table 3.73 Register Contents of IPGPMTUM2**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 5 | W1 | This bit sets whether to enable write access to IPIR, MECNT, and COMPTEST.<br>0: Write access to IPIR MECNT, and COMPTEST is treated as violation. (Value after reset)<br>1: Write access to IPIR MECNT, and COMPTEST is not restricted. |
| 4 | R1 | This bit sets whether to enable read access to IPIR, MECNT, and COMPTEST.<br>0: Read access to IPIR MECNT, and COMPTEST is treated as violation. (Value after reset)<br>1: Read access to IPIR MECNT, and COMPTEST is not restricted. |
| 3, 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | W0 | This bit sets whether to enable write access to INTC1.<br>0: Write access to INTC1 is treated as violation. (Value after reset)<br>1: Write access to INTC1 is not restricted. |
| 0 | R0 | This bit sets whether to enable read access to INTC1.<br>0: Read access to INTC1 is treated as violation. (Value after reset)<br>1: Read access to INTC1 is not restricted. |

(g) IPGPMTUM3 — Peripherals Protection Setting Register 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | W1 | R1 | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R | R | R |

**Table 3.74 Register Contents of IPGPMTUM3**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 5 | W1 | This bit sets whether to enable write access to SysErrGen.<br>0: Write access to SysErrGen is treated as violation. (Value after reset)<br>1: Write access to SysErrGen is not restricted. |
| 4 | R1 | This bit sets whether to enable read access to SysErrGen.<br>0: Read access to SysErrGen is treated as violation. (Value after reset)<br>1: Read access to SysErrGen is not restricted |
| 3 to 0 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |

(h) IPGPMTUM4 — Peripherals Protection Setting Register 4

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | X1 | W1 | R1 | — | — | W0 | R0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R | R | R/W | R/W |

**Table 3.75 Register Contents of IPGPMTUM4**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | Reserved. The bit is always read as 0. The write value should always be 0. |
| 6 | X1 | This bit sets whether to enable instruction fetch read access to peripherals to be connected to the H-Bus.<br>0: Instruction fetch read access to peripherals to be connected to the H-Bus is treated as violation. (value after reset)<br>1: Instruction fetch read access to peripherals to be connected to the H-Bus is not restricted. |
| 5 | W1 | This bit sets whether to enable write access to peripherals to be connected to the H-Bus.<br>0: Write access to peripherals to be connected to the H-Bus is treated as violation. (value after reset)<br>1: Write access to peripherals to be connected to the H-Bus is not restricted. |
| 4 | R1 | This bit sets whether to enable read access to peripherals to be connected to the H-Bus.<br>0: Read access to peripherals to be connected to the H-Bus is treated as violation. (Valueafter reset)<br>1: Read access to peripherals to be connected to the H-Bus is not restricted |
| 3 to 2 | Reserved | Reserved. These bits are always read as 0. The write value should always be 0. |
| 1 | W0 | This bit sets whether to enable write access to the PEG.<br>0: Write access to the PEG is treated as violation. (Value after reset)<br>1: Write access to the PEG is not restricted. |
| 0 | R0 | This bit sets whether to enable read access the PEG.<br>0: Read access to the PEG is treated as violation. (Value after reset)<br>1: Read access to the PEG is not restricted |

### 3.2.4.3 System Error Notification Control Function (SEG)

SEG (SysErrGen) controls interrupt notification and recording after a system error occurred by a data access.

Multiple error occurrence inputs are categorized according to error factors, and are processed sequentially from the highest-priority error factor, generating an FE-level asynchronous exception (SYSERR).

The bit position of the SEGFLAG register becomes the priority of error factors. Error factors of lower bits take precedence over error factors of upper bits.

Error information is recorded once regardless of error frequency.

The error with the highest priority of error factor (in case errors occurred simultaneously) is valid. Recorded error information is not overwritten by subsequent errors.

#### (1) List of SEG Function Control Registers

Table 3.76 SEG Register (Base Address: FFFE E980$_H$)

| Address Offset | Size (Byte) | Register Name | Abbreviation | Right | R/W | Operable Bit | | | | Value after reset |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 1 | 8 | 16 | 32 | |
| +00$_H$ | 2 | Error notification control register | SEGCONT | — | R/W *1 | — | — | √ | — | 0000$_H$ |
| +02$_H$ | 2 | Error occurrence retention register | SEGFLAG | — | R/W *1 | — | — | √ | — | 0000$_H$ |
| +04$_H$ | 2 | Error factor retention register | SEGTYPE | — | R/W *1 | — | — | √ | — | Undefined |
| +06$_H$ | 2 | Error factor retention register (Operation source) | SEGSIDE | — | R/W *1 | — | — | √ | — | Undefined |
| +08$_H$ | 4 | Error factor retention register (address) | SEGADDR | — | R/W *1 | — | — | √ | √ | Undefined |

Note 1. Write accesses from user mode are ignored.

**NOTE**

- If an access is made with an address offset or operable bits other than those specified above, an error response is returned.
- Write access is only possible with the SV privilege. Attempting to write, if these conditions do not hold, leads to an error response being returned.
- No restriction is provided for read accesses.
  — Read accesses to ranges permitted by other protection systems are enabled at any time.

### (2) Register Set

#### (a) SEGCONT — Error Notification Control Register

- This register is used to enable (= 1) or disable (= 0) notification of SysErr request in response to error flags that store error occurrence status according to factors.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | VPGE | VCRE | — | TCME | — | VCIE | — | ICCE | — | NEE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R/W | R | R/W | R | R/W | R | R/W |

**Table 3.77 SEGCONT register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 9 | VPGE | This bit notifies an error in writing access to P-Bus.<br>The error includes the followings:<br>• P-Bus guard error<br>• Address EDC error<br>• Data ECC error<br>• Access to unimplemented area in the P-Bus<br>• P-Bus slave peripherals error |
| 8 | VCRE | This bit notifies the IPG violation access detection and subsequent access blocking.*1 |
| 7 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 6 | TCME | This bit notifies an error in accessing data to its own local RAM.<br>The error includes the following cases:<br>• ECC error<br>• Access to the RAM-unimplemented area in the local RAM |
| 5 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |

**Table 3.77 SEGCONT register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 | VCIE | This bit notifies an error<br>The error includes the followings:<br>• in reading access to P-Bus<br>– P-Bus guard error<br>– Address EDC error<br>– Data ECC error<br>– access to unimplemented area in the P-Bus<br>– P-Bus slave peripheral error<br>• in access to H-Bus<br>– H-Bus guard error<br>– access to unimplemented area in the H-Bus area<br>– H-Bus slave peripheral error<br>• in access to Code Flash<br>– Code Flash guard error<br>– Address Parity error<br>– Data ECC error<br>• in access to Global RAM<br>– GRAM guard error<br>– Address ECC error<br>– Data ECC error<br>• in access to foreign Local RAM<br>– PE guard error<br>– Address ECC error<br>– Data ECC error<br>• in access to system peripheral<br>– access to the unimplemented area in the system peripheral area<br>• This bit notifies the IPG violation access detection and subsequent access blocking.*1<br>• in access privilege violation<br>– n reading or writing access to IPG Protection Setting Registers by user mode (PSW.UM = 1)<br>– In writing access to the SEG Function Control Registers by user mode (PSW.UM = 1) |
| 3 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 2 | ICCE | Instruction cache error notification enable<br>The error that occurred in the instruction cache is handled when the instruction cache system register, ICCTRL.ICHEMK, is set to 0 (whose value after reset is 1):<br>For instruction cache errors, see **(g) ICERR — Instruction cache error register** in **Section 3.2.1.2, Register Set**, **(6) Cache Operation Function registers**. |
| 1 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 0 | NEE | This bit notifies the correctable ECC error<br>Code Flash, Global RAM, Local RAM, Foreign Local RAM<br>To enable this function, it is necessary to set VCIE or TCME bit at the same time. |

Note 1. For error factor address, see **Section 3.2.4.2, PE's Internal Peripherals Protection Function (IPG)**, IPGADRUM register.

(b) SEGFLAG — Error Occurrence Retention Register

- The register SEGFLAG indicates from which slaves error responses have been received so far. SEGFLAG is not cleared automatically. Clearing can be performed by writing a zero into the register.
- Writing to the register enables both setting and clearing.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | VPGF | VCRF | — | TCMF | — | VCIF | — | ICCF | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R/W | R | R/W | R | R/W | R | R |

**Table 3.78 SEGFLAG register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 9 | VPGF | Flag corresponding to bit 9 of the SEGCONT register |
| 8 | VCRF | Flag corresponding to bit 8 of the SEGCONT register |
| 7 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 6 | TCMF | Flag corresponding to bit 6 of the SEGCONT register |
| 5 | Reserved | When read, the value is undefined. When writing, always write the value after reset. |
| 4 | VCIF | Flag corresponding to bit 4 of the SEGCONT register |
| 3 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 2 | ICCF | Flag corresponding to bit 2 of the SEGCONT register |
| 1, 0 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |

**Table 3.79 Relationship between System Error Exception Cause Code and SEGFLAG**

| FEIC | Error Cause |
|---|---|
| $10_H$ | Reserved |
| $11_H$ | Instruction fetch from code flash |
| $12_H$ | SEGFLAG.ICCF |
| $13_H$ | Instruction fetch from other than code flash |
| $14_H$ | SEGFLAG.VCIF |
| $15_H$ | Reserved |
| $16_H$ | SEGFLAG.TCMF |
| $17_H$ | Reserved |
| $18_H$ | SEGFLAG.VCRF |
| $19_H$ | SEGFLAG.VPGF |
| $1A_H$ to $1F_H$ | Reserved |

(c) Error Factor Retention Register (SEGTYPE)

This register records information of an error factor that notifies a SYSERR request (only one history is recorded). It records ETYPE[6:0] and CODE[3:0] where the error factors for the VPGF, VCIF and TCMF bits of the SEGFLAG register were found. The error factors except VPGF, VCIF and TCMF bits of the SEGFLAG register to be recorded as ETYPE[6:0] = $0000000_B$ and CODE[3:0] = $0000_B$. This can not be modified when the error occurrence flag to enable notification is set.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETYPE[6:0] | | | | | | | — | — | — | — | — | CODE[3:0] | | | |
| Value after reset | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 3.80 SEGTYPE register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 9 | ETYPE[6:0] | Error Type<br>6: Address Error<br>5: Response Error<br>4: Uncorrected Error<br>3: Protect Error<br>2: Request Error<br>1: Un-support Error<br>0: Boundary Error |
| 8 to 4 | Reserved | The read value is undefined. The write value should always be 0. |
| 3 to 0 | CODE[3:0] | Error Cause Code<br>These bits indicate bit position in SEGFLAG. |

(d) SEG Error Information (Master) Register (SEGSIDE)

This register records information of an error factor that notifies a SYSERR request (only one history is recorded). It records PEID[2:0] where the error factors for the VPGF, VCIF, VCRF and TCMF bits of the SEGFLAG register were found. The error factors except VPGF, VCIF, VCRF and TCMF bits of the SEGFLAG register to be recorded as PEID[2:0] = $000_B$. It records SPID[4:0] where the error factors for the VCIF and TCMF bits of the SEGFLAG register were found.

The error factors except VCIF and TCMF bits of the SEGFLAG register to be recorded as SPID[4:0] = $00000_B$. It records UM where the error factors for the VCIF, VCRF and TCMF bits of the SEGFLAG register were found. The error factors except VCIF, VCRF and TCMF bits of the SEGFLAG register to be recorded as UM = $0_B$. This can not be modified when the error occurrence flag to enable notification is set.

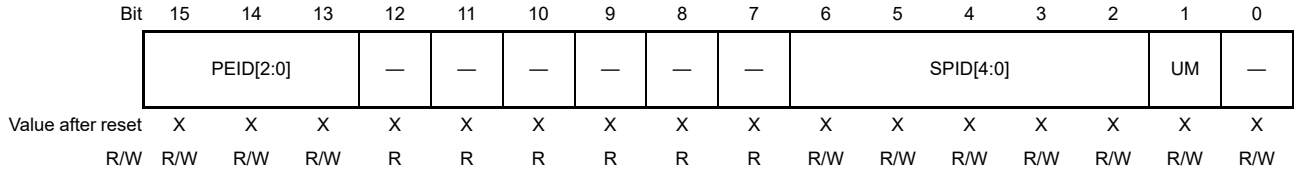



**Table 3.81 SEGSIDE register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 13 | PEID[2:0] | PEID of the access source from which the SYSERR is generated. |
| 12 to 7 | Reserved | The read value is undefined. The write value should always be 0. |
| 6 to 2 | SPID[4:0] | SPID of the access source from which the SYSERR is generated. |
| 1 | UM | UM of the access source from which the SYSERR is generated |
| 0 | Reserved | The read value is undefined. The write value should always be 0. |

### (e) SEGADDR — SEG Error Information (Address) Register

This register records information of an error factor that notifies a SysErr request (only one history is recorded). It records the addresses where the error factors for the VCIF and TCMF bits of the SEGFLAG register were found. The error factors except VCIF and TCMF bits of the SEGFLAG register to be recorded as 0000 $0000_H$. This can not be modified when the error occurrence flag to enable notification is set.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Address[31:16] | | | | | | | | | | | | | | | |
| Value after reset | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| R/W | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Address[15:0] | | | | | | | | | | | | | | | |
| Value after reset | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| R/W | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 | R/W*1 |

×: Undefined (retained)

Note 1. This can not be modified when the error occurrence flag to enable notification is set.

**Table 3.82 SEGADDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | Address[31:0] | Request address of operation source |

### (3) SEG Function

#### (a) SEG function: Notifying a SysErr request due to an error flag

- Setting an error flag takes precedence over clearing the same flag.
  - Simultaneous clearing operation is ignored.
- Priority of error factors
  - The bit position of the notification-enabled SEGFLAG register becomes the priority of error factors. Error factors of lower bits take precedence over error factors of upper bits. Notification is made from the highest-priority error factor.
  - The bit position of error factors is notified as a “SysErr factor code.”
- Conditions for starting SysErr request notification
  - Even if a SEGFLAG register bit is set to 1, notification is not made.
  - Notification is made immediately after a SEGCONT register bit is set to 1.
  - After clearing SEGFLAG register bits of error notification subject, notification is made depending on priority of SEGFLAG register bits except for cleared bits (re-arbitration).
- Finishing notification at a SysErr request response
  - Even after notification is finished, the flag is not cleared automatically.
  - Notification is not made until re-arbitration is performed by setting or clearing the flag.
  - If an error flag that is prioritized higher than the error factor is set prior to a request response, the notification information may be replaced with an upper SysErr factor code.

(b) SEG function: Recording error factor information

- When notification-enabled error occurrence is input, the error address is retained in the above register.
  - No information is retained by setting or clearing an error flag described in "**(3) (a) SEG function: Notifying a SysErr request due to an error flag**" above.
  - When multiple error occurrence inputs are present simultaneously, information other than the prioritized error factor is not retained.
- While the notification-enabled error flag described in "**(3) (a) SEG function: Notifying a SysErr request due to an error flag**" above is set to 1, overwrite to the above register is inhibited
  - If error occurrence input continues, information of subsequent error factors is not retained.
  - To reset the inhibition of overwrite to the register, clear either SEGCONT or SEGFLAG register (or both of them).

(c) Additional information on SYSERR exception

- Even if an SYSERR exception occurs, the value of the PSW.EBV bit is kept and the base address of an exception handler does not change.
- Error detection in instruction cache
  Even if an error is detected in instruction cache, resumable SYSERR exception by instruction fetch factor does not occur. Instruction cache automatically invalidates the entry which includes an error and the CPU continues the instruction execution by refetching from the code flash. When the ICCTRL.ICHEMK bit in the system register is set to 0, an error occurred in instruction cache is notified to the SEG.

### 3.2.4.4 FLI (Code-Flash) Guard

The code flash has its own guard. If an illegal access occurs, the code flash guard uses three registers to report information about the violation. The register MGDCFSTAT indicates if an access violation has occurred. The register MGDCFTYPE describes the access type of the first access that triggered a violation. Finally, the register MGDCFCTL is used to reset the MGDCFSTAT register.

#### (1) List of FLI Guard Control Registers

Table 3.83 FLI Guard Register (Base Address: FFC4 $8000_H$)

| Address Offset | Size (Byte) | Register Name | Abbreviation | PBG | R/W | Operable Bit 1 | 8 | 16 | 32 | Value after reset |
|---|---|---|---|---|---|---|---|---|---|---|
| $+100_H$ | 4 | FLI Guard Control Register (SIC) | MGDCFCTL_VCI2CFB | APBGRD_PFSS1.SP4 | R/W | — | √ | √ | √ | 0000 $0000_H$ |
| $+104_H$ | 4 | FLI Guard Error Status Register (SIC) | MGDCFSTAT_VCI2CFB | APBGRD_PFSS1.SP4 | R | — | √ | √ | √ | 0000 $0000_H$ |
| $+10C_H$ | 4 | FLI Guard Error Access Type Register (SIC) | MGDCFTYPE_VCI2CFB | APBGRD_PFSS1.SP4 | R | — | √ | √ | √ | 0000 $0000_H$ |
| $+200_H$ | 4 | FLI Guard Control Register (PE1) | MGDCFCTL_PE1 | APBGRD_PFSS1.SP1 | R/W | — | √ | √ | √ | 0000 $0000_H$ |
| $+204_H$ | 4 | FLI Guard d Error Status Register (PE1) | MGDCFSTAT_PE1 | APBGRD_PFSS1.SP1 | R | — | √ | √ | √ | 0000 $0000_H$ |
| $+20C_H$ | 4 | FLI Guard Error Access Type Register (PE1) | MGDCFTYPE_PE1 | APBGRD_PFSS1.SP1 | R | — | √ | √ | √ | 0000 $0000_H$ |
| $+300_H$ | 4 | FLI Guard Control Register (PE2) | MGDCFCTL_PE2 | APBGRD_PFSS1.SP2 | R/W | — | √ | √ | √ | 0000 $0000_H$ |
| $+304_H$ | 4 | FLI Guard Error Status Register (PE2) | MGDCFSTAT_PE2 | APBGRD_PFSS1.SP2 | R | — | √ | √ | √ | 0000 $0000_H$ |
| $+30C_H$ | 4 | FLI Guard Error Access Type Register (PE2) | MGDCFTYPE_PE2 | APBGRD_PFSS1.SP2 | R | — | √ | √ | √ | 0000 $0000_H$ |

### (2) Register Set

(a) FLI Guard Control Register (VCI2CFB, PE1, PE2) (MGDCFCTL_{VCI2CFB, PE1, PE2})

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLRO | CLRE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 3.84 MGDCFCTL_{VCI2CFB, PE1, PE2} register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | CLRO | Clear the OVF bit of MGDCFSTAT_PE1/PE2 by writing this bit to “1”.<br>Read value:<br>0: Clear is completed<br>1: Clear is on execution |
| 0 | CLRE | Clear the ERR bit of MGDCFSTAT_PE1/PE2 by writing this bit to “1”<br>Read value:<br>0: Clear is completed<br>1: Clear is on execution |

| CLRO | CLRE | Function |
|---|---|---|
| 0 | 0 | Not Clear both bit |
| 0 | 1 | Not Clear both bit (This setting is ignore) |
| 1 | 0 | OVF bit is Cleared |
| 1 | 1 | Both bit is Cleared |

(b) FLI Guard Error Status Register (VCI2CFB, PE1, PE2) (MGDCFSTAT_{VCI2CFB, PE1, PE2})

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF | ERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 3.85 MGDCFSTAT_{VCI2CFB, PE1, PE2} register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | OVF | Error Overflow Status Flag<br>0: No Overflow<br>1: Error Overflow |
| 0 | ERR | Error Status Flag<br>0: No Error<br>1: Error |

(c) FLI Guard Error Access Type Register (VCI2CFB, PE1, PE2) (MGDCFTYPE_{VCI2CFB, PE1, PE2})

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ERRCAUSE | — | — | — | — | — | — | — | — | — | — | — | CFIFADRP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CFPROTTEST | CFPROTAUTHMODE[1:0] | | — | TESTODE | BFA | EXA | CFIFMID[3:0] | | | | — | CFIFTID[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 3.86 MGDCFTYPE_{VCI2CFB, PE1, PE2} register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 28 | ERRCAUSE | This bit indicates if error cause was guard or illegal access<br>0: guard error<br>1: illegal access error |
| 27 to 17 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 16 | CFIFADRP | Address Parity at Guard / illegal access error was occurred |
| 15 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 14 | CFPROTTEST | CFPROTTEST signal at Guard / illegal access error was occurred |
| 13, 12 | CFPROTAUTHMODE[1:0] | CFPROTAUTHMODE signal at Guard / illegal access error was occurred |
| 11 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 10 | TESTMODE | CCIB_test_mode signal at Guard / illegal access error was occurred |
| 9 | BFA | CCIB_BFAA signal at Guard / illegal access error was occurred |
| 8 | EXA | VCI2CFB/PE1_CFV_EXA signal at Guard / illegal access error was occurred |
| 7 to 4 | CFIFMID[3:0] | MID (Master ID) at Guard / illegal access error was occurred<br>MID shows hardware ID status, and it's the following fixed value. This value is captured in each master when the error was occurred.<br>• PE1 = 4'b0001<br>• PE2 = 4'b0010<br>• VCI2CFB = 4'b0111 |
| 3 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 2 to 0 | CFIFTID[2:0] | Transfer Request Type* at Guard / illegal access error was occurred<br>000: Instruction fetch request<br>001: Instruction pre-load request<br>010: Data access request<br>Others: Reserved |

### 3.2.4.5 P-Bus Un-implemented Area Error

If an address is accessed where no peripheral register is implemented, a signal is forwarded to the ECM and the bus access is terminated with an error response. The access context can be read out via the registers above.

#### (1) List of P-Bus Un-implemented Area Access Error Control Registers

Table 3.87 P-Bus Unimplemented Area Error Control Register (Base Address: FFC4C000)

| Address Offset | Size (Byte) | Register Name | Abbreviation | PBG | R/W | Operable Bit | | | | Value after reset |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 1 | 8 | 16 | 32 | |
| $+880_H$ | 4 | ERRSLV Control Register for PFSS P-BUS Area Error | ERRSLVCTL_ PBAREA | APBGRD_ PFSS1.SP4 | R/W | — | √ | √ | √ | $0000\ 0000_H$ |
| $+884_H$ | 4 | ERRSLV Status Register for PFSS P-BUS Area Error | ERRSLVSTAT_ PBAREA | APBGRD_ PFSS1.SP4 | R | — | √ | √ | √ | $0000\ 0000_H$ |
| $+88C_H$ | 4 | ERRSLV Error Transfer Type Register for PFSS P-BUS Area Error | ERRSLVTYPE_ PBAREA | APBGRD_ PFSS1.SP4 | R | — | — | √ | √ | $0000\ 0000_H$ |

#### (2) Register Set

(a) ERRSLVCTL_PBAREA — ERRSLV Control Register for PFSS P-Bus Area Error

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLRO | CLRE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

Table 3.88 ERRSLVCTL_PBAREA register contents

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | CLRO | Clear the OVF bit of PGERRSTAT_PE1/PE2 by writing this bit to “1”.<br>Read value:<br>0: Clear is completed<br>1: Clear is on execution |
| 0 | CLRE | Clear the ERR bit of PGERRSTAT_PE1/PE2 by writing this bit to “1”<br>Read value:<br>0: Clear is completed<br>1: Clear is on execution |

(b) ERRSLVSTAT_PBAREA — ERRSLV Status Register for PFSS P-Bus Area Error

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF | ERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 3.89 ERRSLVSTAT_PBAREA**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | OVF | Error Overflow Status Flag<br>0: No Overflow<br>1: Error Overflow |
| 0 | ERR | Error Status Flag<br>0: No Error<br>1: Error |

(c) ERRSLVTYPE_PBAREA — ERRSLV Error Transfer Type Register for PFSS P-Bus Area Error

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | SPID[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID[2:0] | | | — | — | — | — | — | — | UM | — | STRB[3:0] | | | | WRITE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 3.90 ERRSLVTYPE_PBAREA**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. |
| 20 to 16 | SPID[4:0] | SPID at P-Bus Guard error was occurred |
| 15 to 13 | PEID[2:0] | PEID at P-Bus Guard error was occurred |
| 12 to 8 | Reserved | When read, the value after reset is read. |
| 7 | Reserved | When read, the value after reset is read. |
| 6 | UM | UM at P-Bus Guard error was occurred |
| 5 | Reserved | When read, the value after reset is read. |
| 4 to 1 | STRB[3:0] | PSTRB signal at P-Bus Guard error was occurred |
| 0 | WRITE | PWRITE signal at P-Bus Guard error was occurred |

### 3.2.4.6 Checker Core

The PE1, PE2 has the Checker Core for safety assurance, resulting in a highly reliable system. Monitoring the outputs from the PE1, PE2 and the Checker Core with the comparator all the time enables immediate detection of the PE1, PE2 abnormal operations. See **Section 28, Functional Safety** for detail.

# 3.3 Inter CPU Functions Overview

## 3.3.1 Processor Element Identifier

The PEID, each processor element ID number, can be read from the PEID field in the HTCFG0 register. Which CPU core performs a specific program can be understood by referring to the PEID. The following shows the PEID of this product.

| CPUcore | PEID |
|---|---|
| CPU1 (PE1) | 001B |
| CPU2 (PE2) | 010B |

## 3.3.2 Inter-Processor Interrupt Function

Both CPU1 and CPU2 individually have IPIR registers as their own peripheral functions. Setting of the IPIR register enables an EI-level interrupt request from a PE to another PE. For details, see INTC Chapter.

## 3.3.3 Exclusive Function

The Local RAM, Global RAM, and exclusive control register (MEV) are available as a resource for exclusive control. Atomic operation instructions of LDL/STC, CAXI, SET1, CLR1, and NOT1 can be performed for the Local RAM and the Global RAM. Such instructions cannot be performed to H-BUS area. For the exclusive control register (MEV), CAXI, SET1, CLR1, and NOT1 can be performed.

### 3.3.3.1 Exclusive Control Register (MEV)

This register supports exclusive control for variables shared between PEs (common resources).

(MEV = Mutual Exclusion Variable Register)

- 32-bit MEV register is included.
- 1-, 8-, 16-, and 32-bit accesses are available for each MEV.
- Accesses from CPU1 (PE1) and CPU2 (PE2) can be made.
- Atomic operation instructions of CAXI, SET1, CLR1, and NOT1 can be performed.

CPU1 and CPU2 each have an independent access path for the MEV registers. Therefore, when CPU1 and CPU2 each access different MEV registers, they do not need to wait for access. When they access the same MEV register, however, waiting for access is required.

**Table 3.91 MEV Registers (Base Address: FFFE EC00$_H$) (1/2)**

| Address Offset | Size (Byte) | Register Name | Abbreviation | R/W | Operable Bit 1 | 8 | 16 | 32 | Value after reset |
|---|---|---|---|---|---|---|---|---|---|
| +00$_H$ | 32 | Exclusive Control Register 0 | G0MEV0 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +04$_H$ | 32 | Exclusive Control Register 1 | G0MEV1 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +08$_H$ | 32 | Exclusive Control Register 2 | G0MEV2 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +0C$_H$ | 32 | Exclusive Control Register 3 | G0MEV3 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |

Table 3.91 MEV Registers (Base Address: FFFE EC00$_H$) (2/2)

| Address Offset | Size (Byte) | Register Name | Abbreviation | R/W | Operable Bit 1 | 8 | 16 | 32 | Value after reset |
|---|---|---|---|---|---|---|---|---|---|
| +10$_H$ | 32 | Exclusive Control Register 4 | G0MEV4 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +14$_H$ | 32 | Exclusive Control Register 5 | G0MEV5 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +18$_H$ | 32 | Exclusive Control Register 6 | G0MEV6 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +1C$_H$ | 32 | Exclusive Control Register 7 | G0MEV7 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +20$_H$ | 32 | Exclusive Control Register 8 | G0MEV8 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +24$_H$ | 32 | Exclusive Control Register 9 | G0MEV9 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +28$_H$ | 32 | Exclusive Control Register 10 | G0MEV10 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +2C$_H$ | 32 | Exclusive Control Register 11 | G0MEV11 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +30$_H$ | 32 | Exclusive Control Register 12 | G0MEV12 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +34$_H$ | 32 | Exclusive Control Register 13 | G0MEV13 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +38$_H$ | 32 | Exclusive Control Register 14 | G0MEV14 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +3C$_H$ | 32 | Exclusive Control Register 15 | G0MEV15 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +40$_H$ | 32 | Exclusive Control Register 16 | G0MEV16 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +44$_H$ | 32 | Exclusive Control Register 17 | G0MEV17 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +48$_H$ | 32 | Exclusive Control Register 18 | G0MEV18 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +4C$_H$ | 32 | Exclusive Control Register 19 | G0MEV19 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +50$_H$ | 32 | Exclusive Control Register 20 | G0MEV20 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +54$_H$ | 32 | Exclusive Control Register 21 | G0MEV21 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +58$_H$ | 32 | Exclusive Control Register 22 | G0MEV22 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +5C$_H$ | 32 | Exclusive Control Register 23 | G0MEV23 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +60$_H$ | 32 | Exclusive Control Register 24 | G0MEV24 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +64$_H$ | 32 | Exclusive Control Register 25 | G0MEV25 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +68$_H$ | 32 | Exclusive Control Register 26 | G0MEV26 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +6C$_H$ | 32 | Exclusive Control Register 27 | G0MEV27 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +70$_H$ | 32 | Exclusive Control Register 28 | G0MEV28 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +74$_H$ | 32 | Exclusive Control Register 29 | G0MEV29 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +78$_H$ | 32 | Exclusive Control Register 30 | G0MEV30 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |
| +7C$_H$ | 32 | Exclusive Control Register 31 | G0MEV31 | R/W | √ | √ | √ | √ | 0000 0000$_H$ |

### 3.3.3.2 Operation of the LDL.W and STC.W Instructions

The LDL.W and STC.W instructions can be used to obtain atomic read-modify-write operations for accurate processing in the updating of memory by multicore systems. The LDL.W and STC.W instructions operate as follows. The LDL.W and STC.W instructions are only in the instruction set for the PE1/PE2.

- Link generation: The CPU is capable of generating links to both the local RAM and global RAM. Executing the LDL.W instruction on the target RAM for the operation leads to the link flag being set and a link being generated in response to reading by the instruction. Two link flag systems are provided, one each for the following two areas of RAM.

(1) The local RAM for the given processor

(2) The global RAM

- Success in storing: After a link has been generated, storing will only proceed in response to executing an STC.W instruction corresponding to the generated link.
- Failure in storing: If a link is lost, storing does not proceed even when an STC.W instruction for the corresponding address is processed. Storing also does not proceed when an STC.W instruction that does not correspond to the link is processed.
- Condition for successful storing: If the following condition is met, the STC.W instruction is judged to be for the address corresponding to the link.
  - The address for the LDL.W instruction which generated the link matches that for the STC.W instruction.
- Loss of the link: If any of the following occurs, the link flag is cleared and the link is lost.
  - Any of the following processing by the CPU for which the link was generated:
    - CLL instruction being executed.
    - An STC.W instruction being executed. The corresponding link (for (1) or (2) above) will be lost irrespective of the success or failure of storing.
    - Occurrence of an exception or the CPU executing an instruction to return from an exception processing routine (FERET or EIRET). The link for the linked RAM area and the link flag are cleared.
    - A successive LDL.W instruction being executed for a location corresponding to the link flag for the linked RAM. The link generated in response to the preceding LDL.W instruction will be lost. Do not execute such processing.
    - Storing operations other than execution of a STC.W instruction for the address indicated by the link. Do not execute such processing.
  - Access as described below by another bus master:
    Any storing operation, including execution of an STC.W instruction for the address indicated by the link. The corresponding link will be lost.

Success of the STC.W instruction means that the LDL.W and STS.W instructions have realized an atomic Read-Modify-Write operation.

### 3.3.4 Write-Through Buffer for Global RAM

If global RAM is implemented, it is accessible by each CPU core through a 64-bit bus.

PE1 and PE2 share the common Write-Through Buffer (WT-Buffer or WT-Buf for short). The WT-Buf consists of 2 banks (one for each GRAM bank) and each bank consists of 8 entries x 64bits. The WT-Buf behavior is summarized below according to access type from PE:

Read hit: data are returned from the WT-Buf.

Read miss: data are read from GRAM and they are registered in WT-Buf

Write hit: Both WT-Buf and GRAM are updated

Write miss (no RmW): Both WT-Buf and GRAM are updated

Write miss (with RmW): Read from GRAM first and write the parts to both WT-Buf and GRAM

Since one WT-Buffer is shared for both PE1 and PE2, no coherency control is needed. The data coherency between GRAM and WT-Buffer is always kept because the same data is always written to GRAM and WT-Buffer at the same time. The data coherency between PE and DMA is also kept, as DMA writes to GRAM as well as WT-Buffer at the same time and DMA always reads from GRAM not from WT-Buffer.

The WT-Buffer equips invalidation timer to invalidate the entries automatically. Invalidation timer cycle count can be chosen from 16-cycles / 64-cycles / 256-cycles. Software invalidation is also possible and one register is provided for this purpose.

The WT-Buffer configuration itself can also be chosen either On or Off. After reset, WT-Buffer is set as On.

**Table 3.92 Write-Through Buffer Control Register (Base Address: FFFF 7A00$_H$)**

| Address | Register Name | Description | PBG | Access Width | Value after reset |
|---|---|---|---|---|---|
| <GRAMC_QOSREG_base> + $4_H$ | GRAMC_ WTBCONFIG0 | GRAMC WTBuf Configuration Register 0 | APBGRD_ PFSS1.SP4 | 32bit | $1_H$ |
| <GRAMC_QOSREG_base> + $8_H$ | GRAMC_ WTBCONFIG1 | GRAMC WTBuf Configuration Register 1 | APBGRD_ PFSS1.SP4 | 32bit | $7_H$ |
| <GRAMC_QOSREG_base> + $C_H$ | GRAMC_ WTBCONFIG2 | GRAMC WTBuf Configuration Register 2 | APBGRD_ PFSS1.SP4 | 32bit | $0_H$ |

Note 1. Configuration of above registers are common, i.e. effects both PEs and both GRAM banks

### (1) Register Sets

(a) GRAMC_WTBCONFIG0 — GRAMC WTBuf Configuration Register 0

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | WTBufMode |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 3.93 GRAMC_WTBCONFIG0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 0 | WTBufMode | This bit sets whether to enable WT Buf<br>0: disable WT Buf<br>1: enable WT Buf |

GRAMC_WTBCONFIG0.WTBufMode change is possible under the following conditions.

– All of GRAM masters (PE1/PE2/ICUM/H-Bus master/Debug) must not access to GRAM.

To change WTBufMode, it is recommended to follow the guides shown below from (1) to (5) to prevent from violating the conditions above.

(1) Inform to other masters that PE1 starts WTBufMode change. Until (5) other masters must not access to GRAM.

(2) PE1: Execute DI instruction for interrupt disabling.

(3) Operate (3a) and (3b), (3c) for the necessary WTBufMode settings.

(3a) PE1: Write to GRAMC_WTBCONFIG0 register by the target value for new WTBufMode.
(3b) PE1: Read the GRAMC_WTBCONFIG0 register to confirm that WTBufMode setting was changed.
(3c) PE1: Execute SYNCP instruction.

(4) PE1: Execute EI instruction for interrupt enabling.

(5) Inform to other masters that PE1 completes WTBufMode change.

If it is necessary to confirm data on RAM from PE1 or PE2 while a master accesses to GRAM, the flushing of WT buffer is avairable. Then PE can read data on RAM before flushing.

(b) GRAMC_WTBCONFIG1 — GRAMC WTBuf Configuration Register 1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | WTBufTmrCmp | | WTBufTmrEn |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 3.94 GRAMC_WTBCONFIG1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 2, 1 | WTBufTmrCmp | These bits set WT Buf invalidate timer count number.<br>00: 16 cycles<br>01: 32 cycles<br>10: 64 cycles<br>11: 256 cycles |
| 0 | WTBufTmrEn | This bit sets whether to enable WT Buf invalidation timer.<br>0: disable WT Buf invalidate timer count down<br>1: enable WT Buf invalidate timer count down<br>All WT Buf entries are flushed when this bit is set from 1 to 0. |

(c) GRAMC_WTBCONFIG2 — GRAMC WTBuf Configuration Register 2

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | WTBufTmcClr |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R*1 |

Note 1. The function of this bit is used by read access.

**Table 3.95 GRAMC_WTBCONFIG2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset always is read. |
| 0 | WTBufTmcClr | When this register is read, All WT Buf entries are flushed. The read value of this bit is always 0. |

## 3.4 Usage Notes

### 3.4.1 Synchronization of Store Instruction Completion and Subsequent Instruction Generation

When a control register is updated by a store instruction, there is a time lag between CPU implementation of the store instruction and actual updating of the control register. Therefore, adequate synchronization processing is needed to ensure the control register reflects updated contents before generation of a subsequent instruction. How to perform synchronization processing is shown below.

**When updated results in the control registers are reflected in the implementation of a subsequent instruction:**

This includes the following case: an interrupt is enabled by implementation of an EI instruction after an interrupt request is cleared by access from the control register in the INTC2 and the peripheral circuits. Proceed as follows in this case.

(1) Store instruction to update a control register (ST.W, etc.)

(2) Dummy read of the above-mentioned control register (LD.W etc.)

(3) SYNCP

(4) Subsequent instruction (EI)

Implement the same processing even when the access required after waiting to secure the updating of a given control register (register A) is to another control register (register B). This includes the following cases: the interlinked operation of different peripheral modules and when releasing the interrupt mask in INTC after making peripheral module settings.

However, this processing is unnecessary if control registers A and B are in the same peripheral group.

(1) Issue the instruction for storage to update control register A (ST.W, etc.)

(2) Dummy-read the above control register (LD.W, etc.)

(3) Issue SYNCP.

(4) Issue the instruction to access control register B (ST.W, LD.W, etc.)

The same processing is also required when access to control registers and memory within the scope of protection starts after waiting for the completion of settings for safety functions such as memory protection, ECC checking, and so on.

**When the updated results in the control registers and memories are reflected in the instruction fetch of a subsequent instruction:**

This includes the following case: after writing an instruction in a memory, the written instruction is implemented. Proceed as follows in this case.

(1) Store instruction to update a memory (ST.W, etc.)

(2) Dummy read of the above-mentioned memory (LD.W, etc.)

(3) SYNCP

(4) SYNCI

(5) Subsequent instruction (branch instruction, etc.)

**When SYNCM-instruction is executed after a store instruction from PE1 or PE2:**

SYNCM-instruction waits for the completion of any preceding memory access (excluding the access to HBUS area or instruction fetch) from the PE.

### 3.4.2 Error response handling

The interfaces used to access the different memory and I/O resources are generating error responses in case the access cannot be completed as expected. This includes access to an address where no memory or register is implemented . These error responses can lead to a SYSERR exception that is not resumeable.

### 3.4.3 Speculative accesses handling

The MPU only checks the addresses of the instruction and involved data for the currently executed instruction. Therefore, speculative access to instructions or data (e.g., during a cache prefetch) can only trigger an illegal access if the instruction or data is used during instruction execution. All addresses that are not used by the CPU pipeline are also not checked by the MPU.

### 3.4.4 Timing supervision

The system timer can be used to implement timing supervision. Timing supervision allows detecting special multi-core related problems like deadlocks and starvation of the system. For this purpose, one of the compare registers of the system timer is loaded with the timeout time before entering a critical section. The system timer can also be used to patrol interrupt request rates with little software overhead. The interrupt handler samples the current time by reading the system timer counter, compares it to the previous time and takes corrective action if necessary.

### 3.4.5 Clock Ratio Change

CPU1(PE1) and CPU2(PE2) are working at CLK_CPU while HSB module and peripheral modules are working at CLK_HSB and CLK_LSB, respectively.

Clock gear change can be done by setting control registers of clock controller. See **Section 12, Clock Controller** for details. Clock gear change is possible under the following conditions:

- PE1 and PE2 do not access to H-Bus.
- PE1 and PE2 do not access to Local RAM of other CPU.
- DMA, H-Bus master, ICUMC and Debug (on System Interconnect) do not access to resources inside PE (Local RAM and peripherals inside PE).
- DMA, H-Bus master, ICUMC and Debug (on System Interconnect) do not access to GRAM in case where clock ratio is changed from CLK_CPU = CLK_HSB to CLK_CPU > CLK_HSB.

To change clock ratio, it is recommended to follow the guides shown below (1. – 5.) to prevent from violating the conditions above.

1. Inform to PE2 that PE1 starts clock gear change. Until (5), PE2 must follow the conditions shown above.
2. PE1: Execute DI instruction for interrupt disable.
3. Repeat (3a) and (3b) for the necessary clock divider settings.
   3a. PE1: Write to CLKDxDIV* register by the target value for new clock ratio.
   3b. PE1: Read the following CLKDxSTAT* register to confirm that clock output now corresponds to the actual divider setting in CLKDxDIV*.
4. PE1: Execute EI instruction for interrupt enable.
5. Inform PE2 that PE1 completes clock gear change.

Note: x= 0 and 1

Note that GRAM and CodeFLASH are working at CLK_CPU and accesses to GRAM and CodeFLASH take multiple cycles of CLK_CPU. Therefore, if frequency of CLK_CPU is changed, access time of GRAM and CodeFLASH is also changed.

### 3.4.6 Register Initialization

Some CPU registers are undefined after a HW reset; therefore they may be different in each core. It is recommended to initialize the register before usage to prevent false error indications from the PE if it is running in lockstep mode. The following assembler sequence can be used for this purpose:

```
#ifndef INIT_REGISTER_DISABLE
    -- Initialize CPU register to avoid Compare Unit Mismatch Error
    -- This code must be executed in supervisor mode (PSW.UM = 0)
       mov     r0,     r1
       mov     r0,     r2
       mov     r0,     r3
       mov     r0,     r4
       mov     r0,     r5
       mov     r0,     r6
       mov     r0,     r7
       mov     r0,     r8
       mov     r0,     r9
       mov     r0,     r10
       mov     r0,     r11
       mov     r0,     r12
       mov     r0,     r13
       mov     r0,     r14
       mov     r0,     r15
       mov     r0,     r16
       mov     r0,     r17
       mov     r0,     r18
       mov     r0,     r19
       mov     r0,     r20
       mov     r0,     r21
       mov     r0,     r22
       mov     r0,     r23
       mov     r0,     r24
       mov     r0,     r25
       mov     r0,     r26
       mov     r0,     r27
       mov     r0,     r28
       mov     r0,     r29
       mov     r0,     r30
       mov     r0,     r31

    -- CPU function group system register set
       ldsr    r0,     EIPC
       ldsr    r0,     FEPC
       ldsr    r0,     CTPC
       ldsr    r0,     CTBP
       ldsr    r0,     ASID,2
       ldsr    r0,     EIWR
       ldsr    r0,     FEWR
```

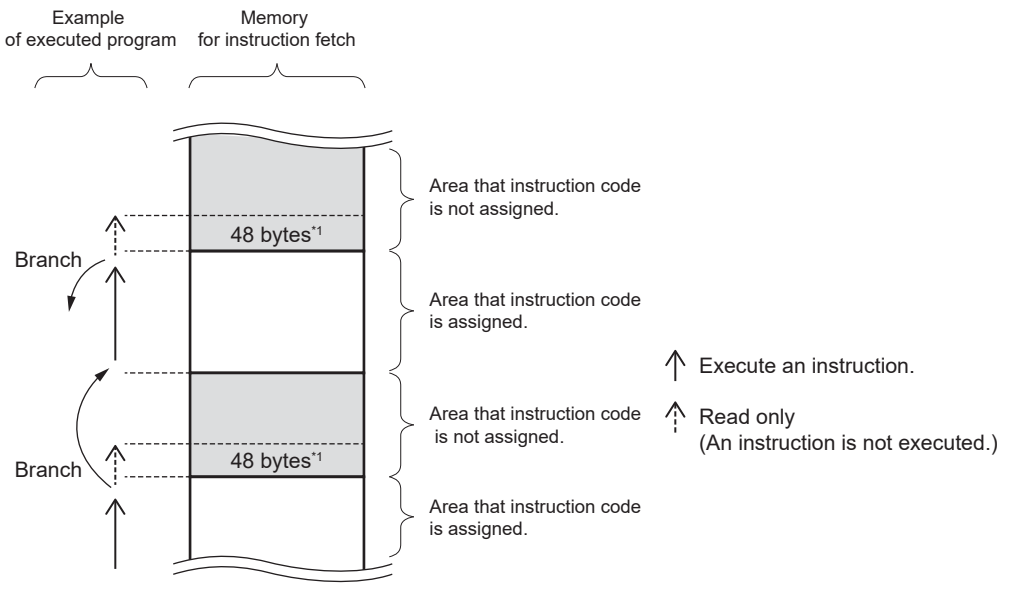
```
    ldsr    r0,     MEA,2
    ldsr    r0,     MEI,2
    ldsr    r0,     EBASE,1
    ldsr    r0,     INTBP,1
    ldsr    r0,     SCCFG,1
    ldsr    r0,     SCBP,1

-- Reserved registers for virtual machine functions
    (Only initialization is necessary)
    ldsr    r0,     sr10, 1
    ldsr    r0,     sr15, 1
    ldsr    r0,     sr13, 1
    ldsr    r0,     sr14, 1

-- FPU registers
    mov     0x00010020,r1
    ldsr    r1,     PSW         -- set FPU usable
    mov     0x00020000,r1
    ldsr    r1,     FPSR
    ldsr    r0,     FPEPC
    ldsr    r0,     FPST
    ldsr    r0,     FPCC

-- SIMD registers (not implemented)

-- MMU registers (not implemented)

-- MPU function registers
    ldsr    r0,     MCA, 5
    ldsr    r0,     MCS, 5
    ldsr    r0,     MCR, 5
    ldsr    r0,     MPLA0, 6
    ldsr    r0,     MPUA0, 6
    ldsr    r0,     MPAT0, 6
    ldsr    r0,     MPLA1, 6
    ldsr    r0,     MPUA1, 6
    ldsr    r0,     MPAT1, 6
    ldsr    r0,     MPLA2, 6
    ldsr    r0,     MPUA2, 6
    ldsr    r0,     MPAT2, 6
    ldsr    r0,     MPLA3, 6
    ldsr    r0,     MPUA3, 6
    ldsr    r0,     MPAT3, 6
    ldsr    r0,     MPLA4, 6
    ldsr    r0,     MPUA4, 6
    ldsr    r0,     MPAT4, 6
    ldsr    r0,     MPLA5, 6
    ldsr    r0,     MPUA5, 6
```

```
    ldsr    r0,     MPAT5, 6
    ldsr    r0,     MPLA6, 6
    ldsr    r0,     MPUA6, 6
    ldsr    r0,     MPAT6, 6
    ldsr    r0,     MPLA7, 6
    ldsr    r0,     MPUA7, 6
    ldsr    r0,     MPAT7, 6
    ldsr    r0,     MPLA8, 7
    ldsr    r0,     MPUA8, 7
    ldsr    r0,     MPAT8, 7
    ldsr    r0,     MPLA9, 7
    ldsr    r0,     MPUA9, 7
    ldsr    r0,     MPAT9, 7
    ldsr    r0,     MPLA10, 7
    ldsr    r0,     MPUA10, 7
    ldsr    r0,     MPAT10, 7
    ldsr    r0,     MPLA11, 7
    ldsr    r0,     MPUA11, 7
    ldsr    r0,     MPAT11, 7
    ldsr    r0,     MPLA12, 7
    ldsr    r0,     MPUA12, 7
    ldsr    r0,     MPAT12, 7
    ldsr    r0,     MPLA13, 7
    ldsr    r0,     MPUA13, 7
    ldsr    r0,     MPAT13, 7
    ldsr    r0,     MPLA14, 7
    ldsr    r0,     MPUA14, 7
    ldsr    r0,     MPAT14, 7
    ldsr    r0,     MPLA15, 7
    ldsr    r0,     MPUA15, 7
    ldsr    r0,     MPAT15, 7

  -- Cache operation function registers
    ldsr    r0,     ICTAGL, 4
    ldsr    r0,     ICTAGH, 4
    ldsr    r0,     ICDATL, 4
    ldsr    r0,     ICDATH, 4
    ldsr    r0,     ICERR, 4

#endif  /* INIT_REGISTER_DISABLE */
```

### 3.4.7 Notes on Prefetch

The CPU performs speculative instruction fetches to areas after the ongoing program area to maintain the instruction fetch throughput. This prefetch may generate a memory read access also from an area (* in **Figure 3.5**) that contains no instruction code. Therefore, note the following. Even if a memory read access from the area (* in **Figure 3.5**) is generated, the read value is not used in instructions. This note is applicable to all memories that allow instruction fetch.

- Occurrence of ECC error due to unstable memory data values
  This prefetch may cause an ECC error in the erased code flash or the uninitialized Local RAM/ Global RAM. When allocating instruction codes in the memory, initialize the relevant area (* in **Figure 3.5**) with desired data.
- Detecting illegal accesses by guard functions
  This prefetch may be detected as an illegal access by the guard functions. To prevent detection of illegal accesses, avoid overlap of the relevant area (* in **Figure 3.5**) with the access prohibited area by guard. Even if memory read access is made to an area protected by the MPU, no memory protection exception occurs.
- Access to access-prohibited area
  Allocate instruction codes in the memory so that the relevant area (* in **Figure 3.5**) does not overlap with access-prohibited areas.




Note 1. An area from instruction code final address to final address + 48 bytes. Even when no instruction code is allocated in this area, memory read may be generated at the time of branching upon a prefetch request.
Note 2. That in case of other memory like LRAM/GRAM the prefetch size is 32 bytes.

**Figure 3.5 Area That Needs Attention for Prefetch**

### 3.4.8 System Register Hazards

To resolve potential hazards when updating the register values of some system registers, implement the following procedures.

- Instruction fetch:

  When fetching instructions after updating the following registers, start to do so only after executing the EIRET instruction, FERET instruction, or a SYNCP instruction followed by SYNCI instruction after executing the register update instruction.

  – PSW.UM, MCFG0.SPID

  When fetching instructions after updating the following registers, start to do so only after executing a SYNCP instruction followed by the SYNCI instruction after executing the register update instruction.

  – ASID, MPU: All related registers (Register number: SR*, 5-7)

- SYSCALL instruction:

  When a SYSCALL instruction is to be executed after updating the register below, execute a SYNCP instruction after the instruction to update the register and before the SYSCALL instruction.

  – SCCFG

- Load/Store:

  When executing instructions that involve load/store operations after updating the following registers, execute the load/store instruction only after executing the SYNCP instruction after executing the register update instruction.

  – ASID, MPU protection area setting registers (Register number: SR*, 6-7)

  Do not execute instructions that involve load/store operations during the instruction preceding and the instruction following update of the following registers.

  – MCTL.MA (For ICUMC only)

- Interrupts:

  Update the following registers in the interrupt disabled state (PSW.ID = 1).

  – PSW.EBV, EBASE, INTBP, ISPR, PMR, ICSR, INTCFG

- Instruction cache clear operation:

  When the confirmation of instruction cache clear operation completed, check the read value of ICCTRL.ICHCLR bit.

- FPU register update:

  When the following register by executing the instruction is updated, execute the SYNCP instruction after executing the instruction.

  – FPU All related registers (Register number: SR6-11, 0)

- FPP/FPI exception mode change:

  When the exception FPP/FPI mode is changed, update the following register after executing the SYNCP and SYNCE instruction. For change the register, apply also the above “FPU register update”.

  – FPSR.PEM

### 3.4.9 Restrictions to use DMAC0/1, DTS ICUM

When DMAC0/1, DTS and ICUM access to the same GRAM Bank0 or Bank1 simultaneously, the number of the masters should be less than or equal to 3 masters and the transfer size of DMAC0/ DMAC1, DTS should be equal or less than 64bit.

### 3.4.10 Restrictions to use H-Bus Master (Ethernet, HS-USRT, FlexRay)

When H-Bus Master transmission/reception data is located to GRAM, the restrictions shown in (1) or (2) is required.

(1) H-Bus Master which can be used simultaneously is up to 2 and respective GRAM Bank have to be separated.
i.e.: Bank0: Ethernet ch0, Bank1: HS-USRT

(2) Either of GRAM Bank0 or Bank1 is only for transmission, and the other is only for reception.
i.e.: Bank0: Transmission data, Bank1: Reception data. However, Ethernet descriptor should be located in LRAM.

## 3.5 Difference among P1M-C, P1H-C and P1H-CE

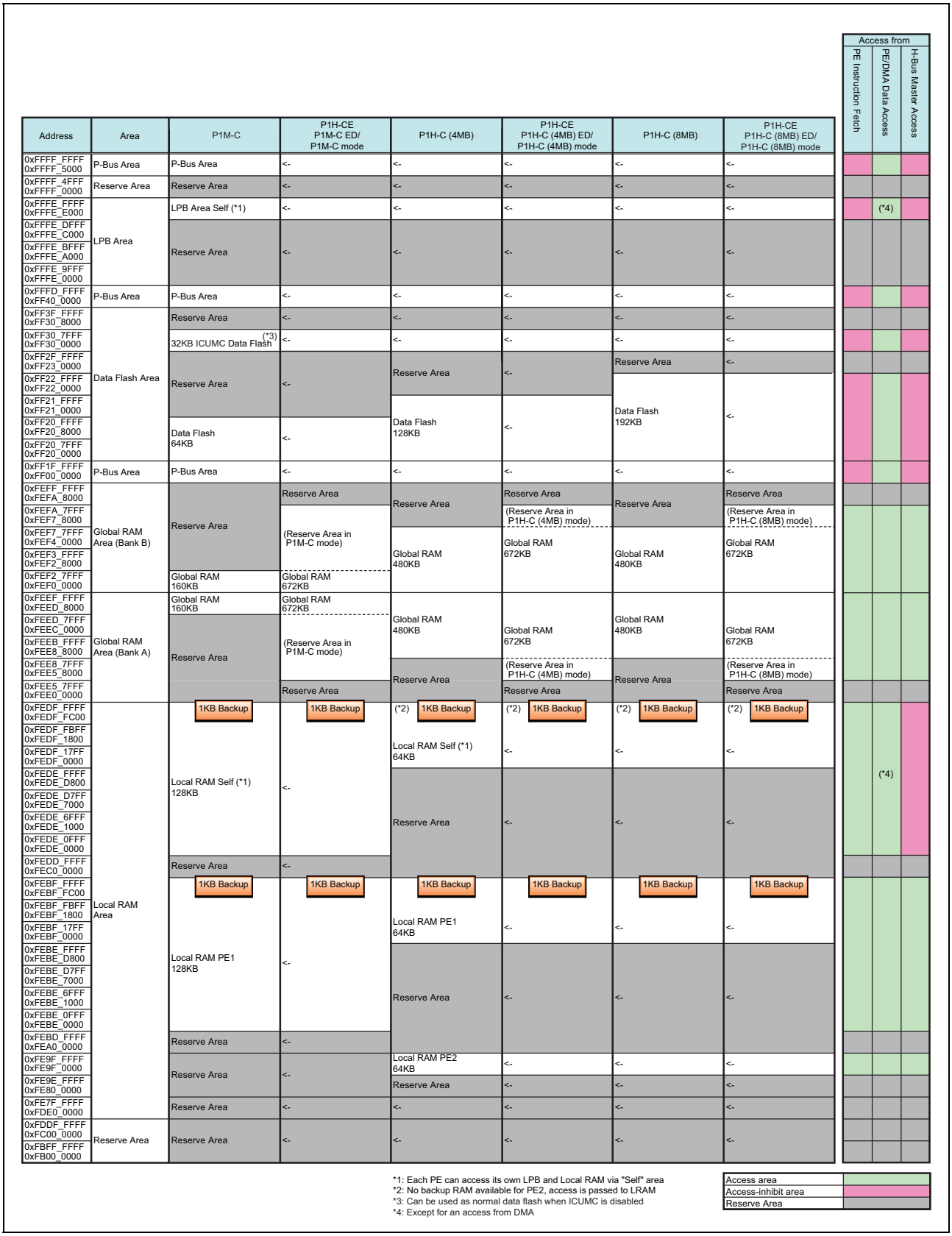
| | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|
| **Number of PEs** | **1** | **2** | **2** | **2** |
| CPU clock freq. [MHz] | 120, 160, 240 | 160, 240 | 160, 240 | 120, 160, 240 |
| Local RAM [KB] | PE1: 128 | PE1: 64<br>PE2: 64 | PE1: 64<br>PE2: 64 | PE1: 128<br>PE2: 64 |
| GlobalRAM [KB] | 320 | 960 | 960 | 1344 |
| IPIR | — | Yes | Yes | Yes |
| Semaphore Registers | — | Yes | Yes | Yes |
| FLI Guard Registers | *1 | Yes | Yes | Yes |

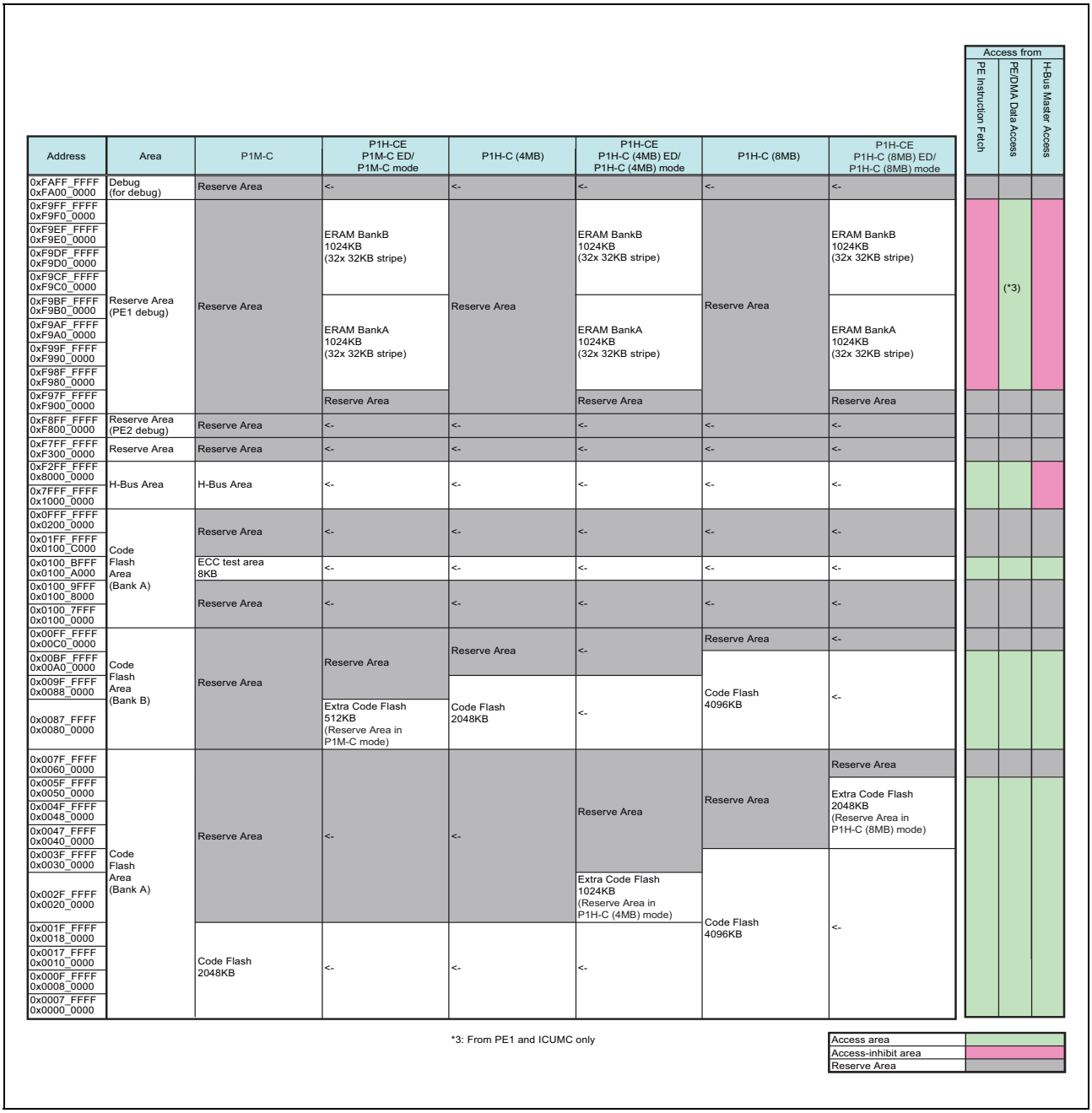
Note 1. The registers for PE2 are disabled.

# Section 4 Address Map

## 4.1 Address Map

**Figure 4.1** shows the address map for each product.


Access from
PE Instruction Fetch
PE/DMA Data Access
H-Bus Master Access
Address
Area
P1M-C
P1H-CE P1M-C ED/ P1M-C mode
P1H-C (4MB)
P1H-CE P1H-C (4MB) ED/ P1H-C (4MB) mode
P1H-C (8MB)
P1H-CE P1H-C (8MB) ED/ P1H-C (8MB) mode
0xFFFF_FFFF 0xFFFF_5000
P-Bus Area
0xFFFF_4FFF 0xFFFF_0000
Reserve Area
0xFFFE_FFFF 0xFFFE_E000
LPB Area
0xFFFE_DFFF 0xFFFE_C000
0xFFFE_BFFF 0xFFFE_A000
0xFFFE_9FFF 0xFFFE_0000
0xFFFD_FFFF 0xFF40_0000
0xFF3F_FFFF 0xFF30_8000
0xFF30_7FFF 0xFF30_0000
0xFF2F_FFFF 0xFF23_0000
0xFF22_FFFF 0xFF22_0000
0xFF21_FFFF 0xFF21_0000
0xFF20_FFFF 0xFF20_8000
0xFF20_7FFF 0xFF20_0000
Data Flash Area
0xFF1F_FFFF 0xFF00_0000
0xFEFF_FFFF 0xFEFA_8000
0xFEFA_7FFF 0xFEF7_8000
0xFEF7_7FFF 0xFEF4_0000
0xFEF3_FFFF 0xFEF2_8000
0xFEF2_7FFF 0xFEF0_0000
Global RAM Area (Bank B)
0xFEEF_FFFF 0xFEED_8000
0xFEED_7FFF 0xFEEC_0000
0xFEEB_FFFF 0xFEE8_8000
0xFEE8_7FFF 0xFEE5_8000
0xFEE5_7FFF 0xFEE0_0000
Global RAM Area (Bank A)
0xFEDF_FFFF 0xFEDF_FC00
0xFEDF_FBFF 0xFEDF_1800
0xFEDF_17FF 0xFEDF_0000
0xFEDE_FFFF 0xFEDE_D800
0xFEDE_D7FF 0xFEDE_7000
0xFEDE_6FFF 0xFEDE_1000
0xFEDE_0FFF 0xFEDE_0000
0xFEDD_FFFF 0xFEC0_0000
0xFEBF_FFFF 0xFEBF_FC00
0xFEBF_FBFF 0xFEBF_1800
0xFEBF_17FF 0xFEBF_0000
0xFEBE_FFFF 0xFEBE_D800
0xFEBE_D7FF 0xFEBE_7000
0xFEBE_6FFF 0xFEBE_1000
0xFEBE_0FFF 0xFEBE_0000
0xFEBD_FFFF 0xFEA0_0000
0xFE9F_FFFF 0xFE9F_0000
0xFE9E_FFFF 0xFE80_0000
0xFE7F_FFFF 0xFDE0_0000
Local RAM Area
0xFDDF_FFFF 0xFC00_0000
0xFBFF_FFFF 0xFB00_0000
Reserve Area
Reserve Area
LPB Area Self (*1)
Reserve Area
32KB ICUMC Data Flash (*3)
Reserve Area
Data Flash 64KB
Reserve Area
Global RAM 160KB
Global RAM 160KB
Reserve Area
Reserve Area
1KB Backup
Local RAM Self (*1) 128KB
Reserve Area
1KB Backup
Local RAM PE1 128KB
Reserve Area
Reserve Area
Reserve Area
Reserve Area
Reserve Area
Reserve Area
Reserve Area
Reserve Area
(Reserve Area in P1M-C mode)
Global RAM 672KB
(Reserve Area in P1M-C mode)
Reserve Area
1KB Backup
<-
1KB Backup
<-
Reserve Area
Reserve Area
Reserve Area
Reserve Area
Reserve Area
Reserve Area
Reserve Area
Reserve Area



| Address | Area | P1M-C | P1H-CE P1M-C ED/ P1M-C mode | P1H-C (4MB) | P1H-CE P1H-C (4MB) ED/ P1H-C (4MB) mode | P1H-C (8MB) | P1H-CE P1H-C (8MB) ED/ P1H-C (8MB) mode | Access from: PE Instruction Fetch | Access from: PE/DMA Data Access | Access from: H-Bus Master Access |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xFAFF_FFFF<br>0xFA00_0000 | Debug (for debug) | Reserve Area | <- | <- | <- | <- | <- | | | |
| 0xF9FF_FFFF<br>0xF9F0_0000 | Reserve Area (PE1 debug) | Reserve Area | ERAM BankB 1024KB (32x 32KB stripe) | Reserve Area | ERAM BankB 1024KB (32x 32KB stripe) | Reserve Area | ERAM BankB 1024KB (32x 32KB stripe) | | (*3) | |
| 0xF9EF_FFFF<br>0xF9E0_0000 | | | | | | | | | | |
| 0xF9DF_FFFF<br>0xF9D0_0000 | | | | | | | | | | |
| 0xF9CF_FFFF<br>0xF9C0_0000 | | | | | | | | | | |
| 0xF9BF_FFFF<br>0xF9B0_0000 | | | ERAM BankA 1024KB (32x 32KB stripe) | | ERAM BankA 1024KB (32x 32KB stripe) | | ERAM BankA 1024KB (32x 32KB stripe) | | | |
| 0xF9AF_FFFF<br>0xF9A0_0000 | | | | | | | | | | |
| 0xF99F_FFFF<br>0xF990_0000 | | | | | | | | | | |
| 0xF98F_FFFF<br>0xF980_0000 | | | | | | | | | | |
| 0xF97F_FFFF<br>0xF900_0000 | | | Reserve Area | | Reserve Area | | Reserve Area | | | |
| 0xF8FF_FFFF<br>0xF800_0000 | Reserve Area (PE2 debug) | Reserve Area | <- | <- | <- | <- | <- | | | |
| 0xF7FF_FFFF<br>0xF300_0000 | Reserve Area | Reserve Area | <- | <- | <- | <- | <- | | | |
| 0xF2FF_FFFF<br>0x8000_0000 | H-Bus Area | H-Bus Area | <- | <- | <- | <- | <- | | | |
| 0x7FFF_FFFF<br>0x1000_0000 | | | | | | | | | | |
| 0x0FFF_FFFF<br>0x0200_0000 | Code Flash Area (Bank A) | Reserve Area | <- | <- | <- | <- | <- | | | |
| 0x01FF_FFFF<br>0x0100_C000 | | | | | | | | | | |
| 0x0100_BFFF<br>0x0100_A000 | | ECC test area 8KB | <- | <- | <- | <- | <- | | | |
| 0x0100_9FFF<br>0x0100_8000 | | Reserve Area | <- | <- | <- | <- | <- | | | |
| 0x0100_7FFF<br>0x0100_0000 | | | | | | | | | | |
| 0x00FF_FFFF<br>0x00C0_0000 | Code Flash Area (Bank B) | Reserve Area | Reserve Area | Reserve Area | <- | Reserve Area | <- | | | |
| 0x00BF_FFFF<br>0x00A0_0000 | | | | | | Code Flash 4096KB | <- | | | |
| 0x009F_FFFF<br>0x0088_0000 | | | | Code Flash 2048KB | <- | | | | | |
| 0x0087_FFFF<br>0x0080_0000 | | | Extra Code Flash 512KB (Reserve Area in P1M-C mode) | | | | | | | |
| 0x007F_FFFF<br>0x0060_0000 | Code Flash Area (Bank A) | Reserve Area | <- | <- | Reserve Area | Reserve Area | Reserve Area | | | |
| 0x005F_FFFF<br>0x0050_0000 | | | | | | | Extra Code Flash 2048KB (Reserve Area in P1H-C (8MB) mode) | | | |
| 0x004F_FFFF<br>0x0048_0000 | | | | | | | | | | |
| 0x0047_FFFF<br>0x0040_0000 | | | | | | | | | | |
| 0x003F_FFFF<br>0x0030_0000 | | | | | | Code Flash 4096KB | <- | | | |
| 0x002F_FFFF<br>0x0020_0000 | | | | | Extra Code Flash 1024KB (Reserve Area in P1H-C (4MB) mode) | | | | | |
| 0x001F_FFFF<br>0x0018_0000 | | Code Flash 2048KB | <- | <- | <- | | | | | |
| 0x0017_FFFF<br>0x0010_0000 | | | | | | | | | | |
| 0x000F_FFFF<br>0x0008_0000 | | | | | | | | | | |
| 0x0007_FFFF<br>0x0000_0000 | | | | | | | | | | |

*3: From PE1 and ICUMC only

| |
|---|
| Access area |
| Access-inhibit area |
| Reserve Area |

**Figure 4.1 Address Map (P1M-C, P1H-C and P1H-CE) (2/2)**

## 4.2 Address space viewed from each bus master

**Figure 4.1** shows address spaces viewed from each bus master.

### 4.2.1 Space in which instructions can be fetched

1. .Instructions of PE1/PE2 can be fetched from the code flash, its own local RAM and the global RAM.

## 4.3 Error notification for an access to unmapped area

When any master accesses to “unmapped area” in which the significant resource is not assigned, the error notification to ECM or error response via bus to the access master is signaled. Regarding of the reaction for an error response of each master, see **Section 4.3.1** to **Section 4.3.10**.

### 4.3.1 Unmapped Code Flash area access error

In the address region 0000 $0000_H$ to 0FFF $FFFF_H$, when a master accesses to code flash unmapped area, error notification is signaled to ECM (No.72 and No.77) from code flash guard on each path.

### 4.3.2 Unmapped Global RAM area access error

In address region FEE0 $0000_H$ to FEFF $FFFF_H$, when a master accesses to global RAM unmapped area, error notification is signaled to ECM (No.73 and No.78) from global RAM guard on each path.

### 4.3.3 Unmapped Local RAM area access error

For own Local RAM area including self-area, unmapped access error is signaled to the PE as the error response. For other PE’s Local RAM area, PE guard error is signaled to ECM (No.64) and error response is returned to the access master when PE guard is correctly configured. Local RAM area is the below

- Self: FEC0 $0000_H$ to FEDF $FFFF_H$
- PE1: FEA0 $0000_H$ to FEBF $FFFF_H$
- PE2: FE80 $0000_H$ to FE9F $FFFF_H$
- Reserved: FDE0 $0000_H$ to FE7F $FFFF_H$

### 4.3.4 Unmapped Local On-chip I/O area access error

For PE own on-chip I/O area including self-area, an access error to areas where no module is assigned is signaled to the PE as the error response and ECM (No.74). For other PE’s local on-chip I/O area, error response is returned to the access master. Local on-chip I/O area is the below

- Self: FFFE $E000_H$ to FFFE $FFFF_H$
- PE1: FFFE $C000_H$ to FFFE $DFFF_H$*1
- PE2: FFFE $A000_H$ to FFFE $BFFF_H$*2
- Reserved: FFFE $0000_H$ to FFFE $9FFF_H$

**Note 1.** For PE1 and debug master

**Note 2.** For PE2 and debug master

### 4.3.5 Details of P-Bus area

P-Bus on-chip I/O area consists of data flash area "FF20 $0000_H$ to FF3F $FFFF_H$ (Data Flash Area)" and on-chip I/O area "FF00 $0000_H$ to FF1F $FFFF_H$ (P-Bus Area)", "FF40 $0000_H$ to FFFD $FFFF_H$ (P-Bus Area)" and "FFFF $5000_H$ to FFFF $FFFF_H$ (P-Bus Area)". For P-bus on-chip I/O area, an access error to areas where no module is assigned is signaled to the access master as the error response and also ECM (No.75).

**Table 4.1 List of P-bus area access (1/11)**

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) P1H-CE P1M-C mode | P1H-CE P1M-C ED mode | P1M-C (BGA-156) | P1H-C (4MB) P1H-CE P1H-C (4MB) | P1H-C (4MB, BGA-156) | P1H-CE P1H-C (4MB) ED mode | P1H-C (8MB) P1H-CE P1H-C (8MB) | P1H-CE P1H-C (8MB) ED mode | Peripheral Group No. | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FF00 $0000_H$ | FF1E $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #5 | See *Security Hardware Manual* |
| FF1F $0000_H$ | FF1F $FFFF_H$ | System reserved | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FF20 $0000_H$ | FF2F $FFFF_H$ | Data Flash Bank A*1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FF30 $0000_H$ | FF30 $7FFF_H$ | Data Flash Bank B | √ | √ | √ | √ | √ | √ | √ | √ | #5 | See *Security Hardware Manual* |
| FF30 $8000_H$ | FF3F $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FF40 $0000_H$ | FF64 $7FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #1 | ERRSLV1AI |
| FF64 $8000_H$ | FF66 $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #5 | See *Security Hardware Manual* |
| FF67 $0000_H$ | FF67 $3FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #3 | ERRSLV3AI |
| FF67 $4000_H$ | FF67 7FFFH | Unmapped area | — | — | — | — | — | — | — | — | #2 | ERRSLV2AI |
| FF67 $8000_H$ | FF6F $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #1 | ERRSLV1AI |
| FF70 $0000_H$ | FF78 $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #3 | ERRSLV3AI |
| FF79 $0000_H$ | FF8D $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #2 | ERRSLV2AI |
| FF8E $0000_H$ | FF97 $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #1 | ERRSLV1AI |
| FF98 $0000_H$ | FF99 $3FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #4 | ERRSLV4AI |
| FF99 $4000_H$ | FF9F $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #3 | ERRSLV3AI |
| FFA0 $0000_H$ | FFA0 $001F_H$ | FLMD | √ | √ | √ | √ | √ | √ | √ | √ | #4 | ERRSLV4AI |
| FFA0 $0020_H$ | FFA0 $0FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFA0 $1000_H$ | FFA0 $103F_H$ | System reserved | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFA0 $1040_H$ | FFA0 $1FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFA0 $2000_H$ | FFA0 $201F_H$ | System reserved | √ | √ | √ | √ | √ | √ | √ | √ | #4 | ERRSLV4AI |
| FFA0 $2020_H$ | FFA0 $7FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFA0 $8000_H$ | FFA0 $80FF_H$ | FLID | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFA0 $8100_H$ | FFA0 $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFA1 $0000_H$ | FFA1 $2FFF_H$ | FACI0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFA1 $3000_H$ | FFA1 $7FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFA1 $8000_H$ | FFA1 $AFFF_H$ | FACI1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFA1 $B000_H$ | FFA1 $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFA2 $0000_H$ | FFA2 $FFFF_H$ | FACI0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFA3 $0000_H$ | FFA3 $FFFF_H$ | FACI1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFA4 $0000_H$ | FFBA $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFBB $0000_H$ | FFBB $2FFF_H$ | FACI0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFBB $3000_H$ | FFBB $7FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFBB $8000_H$ | FFBB $AFFF_H$ | FACI1 | √ | √ | √ | √ | √ | √ | √ | √ | | |

Table 4.1 List of P-bus area access (2/11)

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) P1H-CE P1M-C mode | P1H-CE P1M-C ED mode | P1M-C (BGA-156) | P1H-C (4MB) P1H-CE P1H-C (4MB) | P1H-C (4MB, BGA-156) | P1H-CE P1H-C (4MB) ED mode | P1H-C (8MB) P1H-CE P1H-C (8MB) | P1H-CE P1H-C (8MB) ED mode | Peripheral Group No. | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFBB $B000_H$ | FFBB $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #4 | ERRSLV4AI |
| FFBC $0000_H$ | FFBC $FFFF_H$ | FACI0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFBD $0000_H$ | FFBD $FFFF_H$ | FACI1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFBE $0000_H$ | FFC0 $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC1 $0000_H$ | FFC1 $7FFF_H$ | PORT | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC1 $8000_H$ | FFC1 $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC2 $0000_H$ | FFC2 $7FFF_H$ | PORT | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC2 $8000_H$ | FFC2 $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $0000_H$ | FFC3 $000F_H$ | DNFA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $0010_H$ | FFC3 $00FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $0100_H$ | FFC3 $010F_H$ | DNFA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $0110_H$ | FFC3 $01FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $0200_H$ | FFC3 $020F_H$ | DNFA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $0210_H$ | FFC3 $02FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $0300_H$ | FFC3 $030F_H$ | DNFA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $0310_H$ | FFC3 $03FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $0400_H$ | FFC3 $040F_H$ | DNFA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $0410_H$ | FFC3 $04FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $0500_H$ | FFC3 $050F_H$ | DNFA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $0510_H$ | FFC3 $05FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $0600_H$ | FFC3 $060F_H$ | DNFA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $0610_H$ | FFC3 $06FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $0700_H$ | FFC3 $070F_H$ | DNFA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $0710_H$ | FFC3 $3FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #4 | ERRSLV4AI |
| FFC3 $4000_H$ | FFC3 $401F_H$ | FLCA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $4020_H$ | FFC3 $40FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $4100_H$ | FFC3 $411F_H$ | FLCA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $4120_H$ | FFC3 $41FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $4200_H$ | FFC3 $421F_H$ | FLCA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $4220_H$ | FFC3 $42FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $4300_H$ | FFC3 $431F_H$ | FLCA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $4320_H$ | FFC3 $43FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $4400_H$ | FFC3 $441F_H$ | FLCA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $4420_H$ | FFC3 $44FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $4500_H$ | FFC3 $451F_H$ | FLCA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $4520_H$ | FFC3 $45FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC3 $4600_H$ | FFC3 $461F_H$ | FLCA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC3 $4620_H$ | FFC4 $7FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 $8000_H$ | FFC4 $80FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | #5 | ERRSLV0AI |
| FFC4 $8100_H$ | FFC4 $810F_H$ | CFGSIC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 $8110_H$ | FFC4 $81FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 $8200_H$ | FFC4 $820F_H$ | CFGPE1 | √ | √ | √ | √ | √ | √ | √ | √ | | |

Table 4.1 List of P-bus area access (3/11)

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) P1H-CE P1M-C mode | P1H-CE P1M-C ED mode | P1M-C (BGA-156) | P1H-C (4MB) P1H-CE P1H-C (4MB) | P1H-C (4MB, BGA-156) | P1H-CE P1H-C (4MB) ED mode | P1H-C (8MB) P1H-CE P1H-C (8MB) | P1H-CE P1H-C (8MB) ED mode | Peripheral Group No. | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFC4 8210$_H$ | FFC4 82FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #5 | ERRSLV0AI |
| FFC4 8300$_H$ | FFC4 830F$_H$ | CFGPE2 | —*2 | √ | — | √ | √ | √ | √ | √ | | |
| FFC4 8310$_H$ | FFC4 8FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 9000$_H$ | FFC4 907F$_H$ | GRGDCFG | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 9080$_H$ | FFC4 90FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 9100$_H$ | FFC4 910F$_H$ | GRGDSIC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 9110$_H$ | FFC4 91FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 9200$_H$ | FFC4 920F$_H$ | GRGDPE1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 9210$_H$ | FFC4 92FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 9300$_H$ | FFC4 930F$_H$ | GRGDPE2 | —*2 | √ | — | √ | √ | √ | √ | √ | | |
| FFC4 9310$_H$ | FFC4 96FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 9700$_H$ | FFC4 970F$_H$ | GRGDHBS | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 9710$_H$ | FFC4 A1FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 A200$_H$ | FFC4 A20F$_H$ | PEGDST1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 A210$_H$ | FFC4 A2FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 A300$_H$ | FFC4 A30F$_H$ | PEGDST2 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 A310$_H$ | FFC4 BFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 C000$_H$ | FFC4 C02F$_H$ | FSGD0A | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 C030$_H$ | FFC4 C03F$_H$ | System reserved | √ | √ | √ | √ | √ | √ | √ | √ | #5 | ERRSLV0AI |
| FFC4 C040$_H$ | FFC4 C07F$_H$ | FSGD0B | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 C080$_H$ | FFC4 C7FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 C800$_H$ | FFC4 C81F$_H$ | ERRSLV0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 C820$_H$ | FFC4 C87F$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 C880$_H$ | FFC4 C88F$_H$ | ERRSLV0AI | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 C890$_H$ | FFC4 C9FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC4 CA00$_H$ | FFC4 CA0F$_H$ | DMACMP | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC4 CA10$_H$ | FFC5 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC5 8000$_H$ | FFC5 803F$_H$ | Reset Controller | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC5 8040$_H$ | FFC5 8FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC5 9000$_H$ | FFC5 900F$_H$ | CCIB0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC5 9010$_H$ | FFC5 93FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC5 9400$_H$ | FFC5 940F$_H$ | System reserved | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC5 9410$_H$ | FFC5 97FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC5 9800$_H$ | FFC5 981F$_H$ | DCIB0 | √ | √ | √ | √ | √ | √ | √ | √ | #5 | See *Security Hardware Manual* |
| FFC5 9820$_H$ | FFC5 98FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC5 9900$_H$ | FFC5 991F$_H$ | DCIB1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC5 9920$_H$ | FFC5 9FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC5 A000$_H$ | FFC5 FFFF$_H$ | See *Security Hardware Manual* | | | | | | | | | #5 | |

Table 4.1 List of P-bus area access (4/11)

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) P1H-CE P1M-C mode | P1H-CE P1M-C ED mode | P1M-C (BGA-156) | P1H-C (4MB) P1H-CE P1H-C (4MB) | P1H-C (4MB, BGA-156) | P1H-CE P1H-C (4MB) ED mode | P1H-C (8MB) P1H-CE P1H-C (8MB) | P1H-CE P1H-C (8MB) ED mode | Peripheral Group No. | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFC6 0000$_H$ | FFC6 03FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #5 | ERRSLV0AI |
| FFC6 0400$_H$ | FFC6 05FF$_H$ | ECCPE1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 0600$_H$ | FFC6 07FF$_H$ | ECCPE2 | —*2 | √ | — | √ | √ | √ | √ | √ | | |
| FFC6 0800$_H$ | FFC6 13FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 1400$_H$ | FFC6 15FF$_H$ | ECCPE1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 1600$_H$ | FFC6 17FF$_H$ | ECCPE2 | —*2 | √ | — | √ | √ | √ | √ | √ | | |
| FFC6 1800$_H$ | FFC6 1FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 2000$_H$ | FFC6 210F$_H$ | ECCFLIC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 2110$_H$ | FFC6 21FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 2200$_H$ | FFC6 23FF$_H$ | ECCSIC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 2400$_H$ | FFC6 25FF$_H$ | ECCPE1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 2600$_H$ | FFC6 27FF$_H$ | ECCPE2 | —*2 | √ | — | √ | √ | √ | √ | √ | | |
| FFC6 2800$_H$ | FFC6 2BFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 2C00$_H$ | FFC6 2DFF$_H$ | BECCFLI | √ | √ | √ | √ | √ | √ | √ | √ | #5 | ERRSLV0AI |
| FFC6 2E00$_H$ | FFC6 33FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 3400$_H$ | FFC6 35FF$_H$ | BECCPE1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 3600$_H$ | FFC6 37FF$_H$ | BECCPE2 | —*2 | √ | — | √ | √ | √ | √ | √ | | |
| FFC6 3800$_H$ | FFC6 3FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 4000$_H$ | FFC6 411F$_H$ | ECCGRC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 4120$_H$ | FFC6 41FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 4200$_H$ | FFC6 43FF$_H$ | ECCSIC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 4400$_H$ | FFC6 45FF$_H$ | ECCPE1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 4600$_H$ | FFC6 47FF$_H$ | ECCPE2 | —*2 | √ | — | √ | √ | √ | √ | √ | | |
| FFC6 4800$_H$ | FFC6 4DFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 4E00$_H$ | FFC6 4FFF$_H$ | ECCHBS | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 5000$_H$ | FFC6 53FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 5400$_H$ | FFC6 561F$_H$ | ECCPE1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 5620$_H$ | FFC6 57FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 5800$_H$ | FFC6 5A1F$_H$ | ECCPE2 | —*2 | √ | — | √ | √ | √ | √ | √ | | |
| FFC6 5A20$_H$ | FFC6 5FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 6000$_H$ | FFC6 61FF$_H$ | BECCPE1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 6200$_H$ | FFC6 63FF$_H$ | BECCPE2 | —*2 | √ | — | √ | √ | √ | √ | √ | | |
| FFC6 6400$_H$ | FFC6 65FF$_H$ | BECCPE1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 6600$_H$ | FFC6 67FF$_H$ | BECCPE2 | —*2 | √ | — | √ | √ | √ | √ | √ | | |
| FFC6 6800$_H$ | FFC6 6FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 7000$_H$ | FFC6 71FF$_H$ | BECCSIC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 7200$_H$ | FFC6 73FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 7400$_H$ | FFC6 75FF$_H$ | BECCSIC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 7600$_H$ | FFC6 77FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 7800$_H$ | FFC6 79FF$_H$ | BECCSIC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 7A00$_H$ | FFC6 7BFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 7C00$_H$ | FFC6 7DFF$_H$ | BECCSIC | √ | √ | √ | √ | √ | √ | √ | √ | | |

Table 4.1 List of P-bus area access (5/11)

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) P1H-CE P1M-C mode | P1H-CE P1M-C ED mode | P1M-C (BGA-156) | P1H-C (4MB) P1H-CE P1H-C (4MB) | P1H-C (4MB, BGA-156) | P1H-CE P1H-C (4MB) ED mode | P1H-C (8MB) P1H-CE P1H-C (8MB) | P1H-CE P1H-C (8MB) ED mode | Peripheral Group No. | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFC6 7E00$_H$ | FFC6 83FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #5 | ERRSLV0AI |
| FFC6 8400$_H$ | FFC6 85FF$_H$ | BECCSIC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 8600$_H$ | FFC6 8DFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 8E00$_H$ | FFC6 91FF$_H$ | BECCSIC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 9200$_H$ | FFC6 9FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 A000$_H$ | FFC6 A1FF$_H$ | BECCSIC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 A200$_H$ | FFC6 BFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC6 C000$_H$ | FFC6 C1FF$_H$ | BECCPBA | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC6 C200$_H$ | FFC6 FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 0000$_H$ | FFC7 001F$_H$ | ECCCSIH0 | √ | √ | √ | √ | √ | √ | √ | √ | #3 | ERRSLV3AI |
| FFC7 0020$_H$ | FFC7 00FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 0100$_H$ | FFC7 011F$_H$ | ECCCSIH2 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC7 0120$_H$ | FFC7 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 1000$_H$ | FFC7 101F$_H$ | ECCTTCAN0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC7 1020$_H$ | FFC7 10FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 1100$_H$ | FFC7 111F$_H$ | ECCMCAN1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC7 1120$_H$ | FFC7 1FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 2000$_H$ | FFC7 201F$_H$ | ECCFLX0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC7 2020$_H$ | FFC7 20FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 2100$_H$ | FFC7 211F$_H$ | ECCFLX0T0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC7 2120$_H$ | FFC7 21FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 2200$_H$ | FFC7 221F$_H$ | ECCFLX0T1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC7 2220$_H$ | FFC7 23FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 2400$_H$ | FFC7 241F$_H$ | ECCFLX1 | — | — | — | √ | —*3 | √ | √ | √ | | |
| FFC7 2420$_H$ | FFC7 24FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 2500$_H$ | FFC7 251F$_H$ | ECCFLX1T0 | — | — | — | √ | —*3 | √ | √ | √ | | |
| FFC7 2520$_H$ | FFC7 25FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 2600$_H$ | FFC7 261F$_H$ | ECCFLX1T1 | — | — | — | √ | —*3 | √ | √ | √ | | |
| FFC7 2620$_H$ | FFC7 3FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 4000$_H$ | FFC7 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #2 | ERRSLV2AI |
| FFC7 8000$_H$ | FFC7 801F$_H$ | ECCCSIH1 | √ | √ | √ | √ | √ | √ | √ | √ | #1 | ERRSLV1AI |
| FFC7 8020$_H$ | FFC7 80FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 8100$_H$ | FFC7 811F$_H$ | ECCCSIH3 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC7 8120$_H$ | FFC7 8FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 9000$_H$ | FFC7 901F$_H$ | ECCMCAN0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFC7 9020$_H$ | FFC7 90FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFC7 9100$_H$ | FFC7 911F$_H$ | ECCMCAN2 | — | — | — | — | — | — | √ | √ | | |
| FFC7 9200$_H$ | FFC9 FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCA 0000$_H$ | FFCA 001F$_H$ | CSIH1 | √ | √ | √ | √ | √ | √ | √ | √ | | |

Table 4.1 List of P-bus area access (6/11)

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) P1H-CE P1M-C mode | P1H-CE P1M-C ED mode | P1M-C (BGA-156) | P1H-C (4MB) P1H-CE P1H-C (4MB) | P1H-C (4MB, BGA-156) | P1H-CE P1H-C (4MB) ED mode | P1H-C (8MB) P1H-CE P1H-C (8MB) | P1H-CE P1H-C (8MB) ED mode | Peripheral Group No. | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFCA 0020$_H$ | FFCA 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #1 | ERRSLV1AI |
| FFCA 1000$_H$ | FFCA 107F$_H$ | CSIH1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCA 1080$_H$ | FFCA 1FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCA 2000$_H$ | FFCA 207F$_H$ | PMMA1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCA 2080$_H$ | FFCA 2FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCA 3000$_H$ | FFCA 301F$_H$ | CSIH3 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCA 3020$_H$ | FFCA 3FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCA 4000$_H$ | FFCA 407F$_H$ | CSIH3 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCA 4080$_H$ | FFCA 4FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCA 5000$_H$ | FFCA 507F$_H$ | PMMA3 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCA 5080$_H$ | FFCA 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCA 8000$_H$ | FFCA 807F$_H$ | HS-USRT1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCA 8080$_H$ | FFCA 8FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #1 | ERRSLV1AI |
| FFCA 9000$_H$ | FFCA 907F$_H$ | HS-USRT3 | — | — | — | √ | —*3 | √ | √ | √ | | |
| FFCA 9080$_H$ | FFCA BFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCA C000$_H$ | FFCA C03F$_H$ | RLN31 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCA C040$_H$ | FFCA CFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCA D000$_H$ | FFCA D03F$_H$ | RLN33 | — | — | — | √ | —*3 | √ | √ | √ | | |
| FFCA D040$_H$ | FFCA FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCB 0000$_H$ | FFCB 007F$_H$ | ECM1 | — | — | — | √ | √ | √ | √ | √ | | |
| FFCB 0080$_H$ | FFCB 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCB 1000$_H$ | FFCB 107F$_H$ | ECM1 | — | — | — | √ | √ | √ | √ | √ | | |
| FFCB 1080$_H$ | FFCB 1FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCB 2000$_H$ | FFCB 207F$_H$ | ECM1 | — | — | — | √ | √ | √ | √ | √ | | |
| FFCB 2080$_H$ | FFCB 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCB 8000$_H$ | FFCB 81FF$_H$ | BECCPB0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCB 8200$_H$ | FFCB FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCC 0000$_H$ | FFCC 007F$_H$ | AURORA | — | √ | — | — | — | √ | - | √ | | |
| FFCC 0080$_H$ | FFCC FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD 0000$_H$ | FFCD 01FF$_H$ | FLASH | √ | √ | √ | √ | √ | √ | √ | √ | #4 | ERRSLV4AI |
| FFCD 0200$_H$ | FFCD 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD 1000$_H$ | FFCD 7FFF$_H$ | System reserved | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCD 8000$_H$ | FFCD 81FF$_H$ | BECCPB4 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCD 8200$_H$ | FFCD 83FF$_H$ | BECCPB5 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCD 8400$_H$ | FFCD 85FF$_H$ | BECCPB6 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCD 8600$_H$ | FFCD 9FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD A000$_H$ | FFCD A03F$_H$ | BIST | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCD A040$_H$ | FFCD BFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD C000$_H$ | FFCD C07F$_H$ | RSENT0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCD C080$_H$ | FFCD 00FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD C100$_H$ | FFCD C17F$_H$ | RSENT1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCD C180$_H$ | FFCD C1FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |

Table 4.1 List of P-bus area access (7/11)

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) P1H-CE P1M-C mode | P1H-CE P1M-C ED mode | P1M-C (BGA-156) | P1H-C (4MB) P1H-CE P1H-C (4MB) | P1H-C (4MB, BGA-156) | P1H-CE P1H-C (4MB) ED mode | P1H-C (8MB) P1H-CE P1H-C (8MB) | P1H-CE P1H-C (8MB) ED mode | Peripheral Group No. | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFCD C200$_H$ | FFCD C27F$_H$ | RSENT2 | √ | √ | √ | √ | √ | √ | √ | √ | #4 | ERRSLV4AI |
| FFCD C280$_H$ | FFCD C2FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD C300$_H$ | FFCD C37F$_H$ | RSENT3 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCD C380$_H$ | FFCD C3FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD C400$_H$ | FFCD C47F$_H$ | RSENT4 | √ | √ | —*3 | √ | —*3 | √ | √ | √ | | |
| FFCD C480$_H$ | FFCD C4FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD C500$_H$ | FFCD C57F$_H$ | RSENT5 | √ | √ | —*3 | √ | —*3 | √ | √ | √ | | |
| FFCD C580$_H$ | FFCD C5FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD C600$_H$ | FFCD C67F$_H$ | RSENT6 | — | — | — | √ | —*3 | √ | √ | √ | | |
| FFCD C680$_H$ | FFCD C6FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD C700$_H$ | FFCD C77F$_H$ | RSENT7 | — | — | — | √ | —*3 | √ | √ | √ | | |
| FFCD C780$_H$ | FFCD C7FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD C800$_H$ | FFCD C87F$_H$ | RSENT8 | — | — | — | — | — | — | — | √ | #4 | ERRSLV4AI |
| FFCD C880$_H$ | FFCD C8FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD C900$_H$ | FFCD C97F$_H$ | RSENT9 | — | — | — | — | — | — | — | √ | | |
| FFCD C980$_H$ | FFCD CEFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD CF00$_H$ | FFCD CF3F$_H$ | RSENT_MDSEL | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCD CF40$_H$ | FFCD DFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCD E000$_H$ | FFCD E01F$_H$ | SINT | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFCD E020$_H$ | FFCD FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFCE 0000$_H$ | FFCF FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #4 | ERRSLV1AI |
| FFD0 0000$_H$ | FFD2 FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #3 | ERRSLV3AI |
| FFD3 0000$_H$ | FFD3 01FF$_H$ | MTTCAN0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD3 0200$_H$ | FFD3 07FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD3 0800$_H$ | FFD3 080F$_H$ | System reserved | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD3 0810$_H$ | FFD3 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD3 1000$_H$ | FFD3 11FF$_H$ | MCAN1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD3 1200$_H$ | FFD3 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD3 8000$_H$ | FFD3 9FFF$_H$ | MTTCAN0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD3 A000$_H$ | FFD3 BFFF$_H$ | MCAN1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD3 C000$_H$ | FFD3 FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD4 0000$_H$ | FFD4 003F$_H$ | DTSCTL | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD4 0040$_H$ | FFD4 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD4 1000$_H$ | FFD4 100F$_H$ | DTSCTL | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD4 1010$_H$ | FFD4 1FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD4 2000$_H$ | FFD4 200F$_H$ | INTCTL | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD4 2010$_H$ | FFD4 2FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD4 3000$_H$ | FFD4 300F$_H$ | INTCTL | — | — | — | √ | √ | √ | √ | √ | | |
| FFD4 3010$_H$ | FFD4 FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD5 0000$_H$ | FFD5 003F$_H$ | DCRB0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD5 0040$_H$ | FFD5 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD5 1000$_H$ | FFD5 103F$_H$ | DCRAB2 | √ | √ | √ | √ | √ | √ | √ | √ | | |

Table 4.1 List of P-bus area access (8/11)

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) P1H-CE P1M-C mode | P1H-CE P1M-C ED mode | P1M-C (BGA-156) | P1H-C (4MB) P1H-CE P1H-C (4MB) | P1H-C (4MB, BGA-156) | P1H-CE P1H-C (4MB) ED mode | P1H-C (8MB) P1H-CE P1H-C (8MB) | P1H-CE P1H-C (8MB) ED mode | Peripheral Group No. | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFD5 1040$_H$ | FFD5 1FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #3 | ERRSLV3AI |
| FFD5 2000$_H$ | FFD5 203F$_H$ | DCRAB4 | — | — | — | √ | √ | √ | √ | √ | | |
| FFD5 2040$_H$ | FFD5 2FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD5 3000$_H$ | FFD5 303F$_H$ | DCRAB6 | — | — | — | √ | √ | √ | √ | √ | | |
| FFD5 3040$_H$ | FFD5 FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD6 0000$_H$ | FFD6 007F$_H$ | ECM0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD6 0080$_H$ | FFD6 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD6 1000$_H$ | FFD6 107F$_H$ | ECM0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD6 1080$_H$ | FFD6 1FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD6 2000$_H$ | FFD6 207F$_H$ | ECM0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD6 2080$_H$ | FFD6 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD6 8000$_H$ | FFD6 803F$_H$ | PIC2 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD6 8040$_H$ | FFD7 FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #3 | ERRSLV3AI |
| FFD8 0000$_H$ | FFD8 001F$_H$ | CSIH0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD8 0020$_H$ | FFD8 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD8 1000$_H$ | FFD8 107F$_H$ | CSIH0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD8 1080$_H$ | FFD8 1FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD8 2000$_H$ | FFD8 207F$_H$ | PMMA0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD8 2080$_H$ | FFD8 2FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD8 3000$_H$ | FFD8 301F$_H$ | CSIH2 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD8 3020$_H$ | FFD8 3FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD8 4000$_H$ | FFD8 407F$_H$ | CSIH2 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD8 4080$_H$ | FFD8 4FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD8 5000$_H$ | FFD8 507F$_H$ | PMMA2 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD8 5080$_H$ | FFD8 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD8 8000$_H$ | FFD8 807F$_H$ | HS-USRT0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD8 8080$_H$ | FFD8 8FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD8 9000$_H$ | FFD8 907F$_H$ | HS-USRT2 | — | — | — | √ | —*3 | √ | √ | √ | | |
| FFD8 9080$_H$ | FFD8 BFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD8 C000$_H$ | FFD8 C03F$_H$ | RLN30 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFD8 C040$_H$ | FFD8 CFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD8 D000$_H$ | FFD8 D03F$_H$ | RLN32 | — | — | — | √ | —*3 | √ | √ | √ | | |
| FFD8 D040$_H$ | FFD8 FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFD9 0000$_H$ | FFDD 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #2 | ERRSLV2AI |
| FFDD 8000$_H$ | FFDD 80FF$_H$ | STM0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFDD 8100$_H$ | FFDD 8FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFDD 9000$_H$ | FFDD 90FF$_H$ | STM1 | — | — | — | √ | √ | √ | √ | √ | | |
| FFDD 9100$_H$ | FFDD CFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFDD D000$_H$ | FFDD D07F$_H$ | FSGD2A | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFDD D080$_H$ | FFDD D0FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFDD D100$_H$ | FFDD D10F$_H$ | ERRSLV2 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFDD D110$_H$ | FFDD D7FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |

Table 4.1 List of P-bus area access (9/11)

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) P1H-CE P1M-C mode | P1H-CE P1M-C ED mode | P1M-C (BGA-156) | P1H-C (4MB) P1H-CE P1H-C (4MB) | P1H-C (4MB, BGA-156) | P1H-CE P1H-C (4MB) ED mode | P1H-C (8MB) P1H-CE P1H-C (8MB) | P1H-CE P1H-C (8MB) ED mode | Peripheral Group No. | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFDD D800$_H$ | FFDD D80F$_H$ | ERRSLV2AI | √ | √ | √ | √ | √ | √ | √ | √ | #2 | ERRSLV2AI |
| FFDD D810$_H$ | FFDF FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFE0 0000$_H$ | FFE7 FFFF$_H$ | GTM0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFE8 0000$_H$ | FFE8 001F$_H$ | ECCGTM0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFE8 0020$_H$ | FFE8 00FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFE8 0100$_H$ | FFE8 011F$_H$ | ECCGTM1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFE8 0120$_H$ | FFE8 01FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFE8 0200$_H$ | FFE8 021F$_H$ | ECCGTM2 | — | — | — | √ | √ | √ | √ | √ | | |
| FFE8 0220$_H$ | FFE8 02FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFE8 0300$_H$ | FFE8 031F$_H$ | ECCGTM3 | — | — | — | √ | √ | √ | √ | √ | | |
| FFE8 0320$_H$ | FFE8 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFE8 8000$_H$ | FFE8 81FF$_H$ | BECCPB1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFE8 8200$_H$ | FFE8 83FF$_H$ | BECCPB2 | √ | √ | √ | √ | √ | √ | √ | √ | #2 | ERRSLV2AI |
| FFE8 8400$_H$ | FFEC FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFED 0000$_H$ | FFED 000F$_H$ | WDTA0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFED 0010$_H$ | FFED 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFED 1000$_H$ | FFED 100F$_H$ | WDTA1 | — | — | — | √ | √ | √ | √ | √ | | |
| FFED 1010$_H$ | FFED 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFED 8000$_H$ | FFED 802F$_H$ | SWD0, CICTL | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFED 8030$_H$ | FFED FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFEE 0000$_H$ | FFEE 003F$_H$ | FSGD1A | √ | √ | √ | √ | √ | √ | √ | √ | #1 | ERRSLV1AI |
| FFEE 0040$_H$ | FFEE 00FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFEE 0100$_H$ | FFEE 010F$_H$ | ERRSLV1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFEE 0110$_H$ | FFEE 01FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFEE 0200$_H$ | FFEE 027F$_H$ | FSGD1B | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFEE 0280$_H$ | FFEE 07FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFEE 0800$_H$ | FFEE 080F$_H$ | ERRSLV1AI | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFEE 0810$_H$ | FFEE FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFEF 0000$_H$ | FFEF 01FF$_H$ | MCAN0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFEF 0200$_H$ | FFEF 07FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFEF 0800$_H$ | FFEF 080F$_H$ | system reserved | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFEF 0810$_H$ | FFEF 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFEF 1000$_H$ | FFEF 11FF$_H$ | MCAN2 | — | — | — | — | — | — | √ | √ | | |
| FFEF 1200$_H$ | FFEF 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFEF 8000$_H$ | FFEF 9FFF$_H$ | MCAN0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFEF A000$_H$ | FFEF BFFF$_H$ | MCAN2 | — | — | — | — | — | — | √ | √ | | |
| FFEF C000$_H$ | FFF6 FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF7 0000$_H$ | FFF7 003F$_H$ | DCRB1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF7 0040$_H$ | FFF7 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF7 1000$_H$ | FFF7 103F$_H$ | DCRB3 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF7 1040$_H$ | FFF7 1FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF7 2000$_H$ | FFF7 203F$_H$ | DCRB5 | — | — | — | √ | √ | √ | √ | √ | | |

Table 4.1 List of P-bus area access (10/11)

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) P1H-CE P1M-C mode | P1H-CE P1M-C ED mode | P1M-C (BGA-156) | P1H-C (4MB) P1H-CE P1H-C (4MB) | P1H-C (4MB, BGA-156) | P1H-CE P1H-C (4MB) ED mode | P1H-C (8MB) P1H-CE P1H-C (8MB) | P1H-CE P1H-C (8MB) ED mode | Peripheral Group No. | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFF7 2040$_H$ | FFF7 2FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | #1 | ERRSLV1AI |
| FFF7 3000$_H$ | FFF7 303F$_H$ | DCRB7 | — | — | — | √ | √ | √ | √ | √ | | |
| FFF7 3040$_H$ | FFF7 FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF8 0000$_H$ | FFF8 FFFF$_H$ | Reset Controller CLMA, CVM | √ | √ | √ | √ | √ | √ | √ | √ | #4 | ERRSLV4AI |
| FFF9 0000$_H$ | FFF9 007F$_H$ | FSGD4A | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 0080$_H$ | FFF9 00FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 0100$_H$ | FFF9 010F$_H$ | ERRSLV4 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 0110$_H$ | FFF9 01FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 0200$_H$ | FFF9 027F$_H$ | FSGD4B | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 0280$_H$ | FFF9 03FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 0400$_H$ | FFF9 07FF$_H$ | See *Security Hardware Manual* | | | | | | | | | #4 | |
| FFF9 0800$_H$ | FFF9 080F$_H$ | ERRSLV4AI | √ | √ | √ | √ | √ | √ | √ | √ | #4 | ERRSLV4AI |
| FFF9 0810$_H$ | FFF9 0FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 1000$_H$ | FFF9 17FF$_H$ | ADCF0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 1800$_H$ | FFF9 1FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 2000$_H$ | FFF9 27FF$_H$ | ADCF1 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 2800$_H$ | FFF9 2FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 3000$_H$ | FFF9 30FF$_H$ | OTS0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 3100$_H$ | FFF9 3FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 4000$_H$ | FFF9 407F$_H$ | FSGD3A | √ | √ | √ | √ | √ | √ | √ | √ | #3 | ERRSLV3AI |
| FFF9 4080$_H$ | FFF9 40FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 4100$_H$ | FFF9 410F$_H$ | ERRSLV3 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 4110$_H$ | FFF9 41FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 4200$_H$ | FFF9 427F$_H$ | FSGD3B | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 4280$_H$ | FFF9 47FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 4800$_H$ | FFF9 480F$_H$ | ERRSLV3AI | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 4810$_H$ | FFF9 7FFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 8000$_H$ | FFF9 81FF$_H$ | BECCPB3 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 8200$_H$ | FFF9 BFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 C000$_H$ | FFF9 C00F$_H$ | NTU0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 C010$_H$ | FFF9 C0FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 C100$_H$ | FFF9 C10F$_H$ | NTU0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 C110$_H$ | FFF9 C1FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 C200$_H$ | FFF9 C20F$_H$ | NTU0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 C210$_H$ | FFF9 C7FF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 C800$_H$ | FFF9 C81F$_H$ | NTU0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 C820$_H$ | FFF9 CBFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFF9 CC00$_H$ | FFF9 CC0F$_H$ | NTU0 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFF9 CC10$_H$ | FFF9 FFFF$_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |

Table 4.1 List of P-bus area access (11/11)

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) P1H-CE P1M-C mode | P1H-CE P1M-C ED mode | P1M-C (BGA-156) | P1H-C (4MB) P1H-CE P1H-C (4MB) | P1H-C (4MB, BGA-156) | P1H-CE P1H-C (4MB) ED mode | P1H-C (8MB) P1H-CE P1H-C (8MB) | P1H-CE P1H-C (8MB) ED mode | Peripheral Group No. | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFFA $0000_H$ | FFFA $01FF_H$ | HBUS guard modules | √ | √ | √ | √ | √ | √ | √ | √ | #3 | ERRSLV3AI |
| FFFA $0200_H$ | FFFA $0FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFFA $1000_H$ | FFFA $10FF_H$ | HSSPID | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFFA $1100_H$ | FFFA $1FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFFA $2000_H$ | FFFA $200F_H$ | HTHDMAC | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFFA $2010_H$ | FFFD $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFFF $5000_H$ | FFFF $77FF_H$ | System reserved | √ | √ | √ | √ | √ | √ | √ | √ | — | — |
| FFFF $7A00_H$ | FFFF $7AFF_H$ | System reserved | √ | √ | √ | √ | √ | √ | √ | √ | #5 | ERRSLV0AI |
| FFFF $7B00_H$ | FFFF $7EFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |
| FFFF $7F00_H$ | FFFF $7FFF_H$ | System reserved | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFFF $8000_H$ | FFFF $80FF_H$ | DMACH | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFFF $8100_H$ | FFFF $83FF_H$ | DMACM | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFFF $8400_H$ | FFFF $AFFF_H$ | DMACH | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFFF $B000_H$ | FFFF $BFFF_H$ | INTC2 | √ | √ | √ | √ | √ | √ | √ | √ | | |
| FFFF $C000_H$ | FFFF $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | | |

Note 1. The size of data flash area depends on device and in case of P1H-CE on the device mode. The other data flash area is treated as Unmapped area.

Note 2. The access does not output an error response in P1H-CE P1MC mode.

Note 3. The access does not output an error response.

### 4.3.6 Details of H-Bus area

H-Bus on-chip I/O area is on-chip I/O area of 1000 $0000_H$ to F2FF $FFFF_H$ in **Table 4.2**. For H-bus on-chip I/O area, an access error to areas where no module is assigned is signaled to the access master as the error response and also ECM (No.76).

**Table 4.2 List of H-Bus area access**

| Start Address | End Address | Module Name | P1M-C (QFP, BGA-292) | P1M-C (BGA-156) | P1H-CE P1M-CE mode, P1H-CE P1M-C mode | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-CE P1H-CE (4MB) mode, P1H-CE P1H-C (4MB) mode | P1H-C (8MB) | P1H-CE P1H-CE (8MB) mode, P1H-CE P1H-C (8MB) mode | Unmapped Area Access Error Information |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1000 $0000_H$ | 1001 $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | ERRSLVHI |
| 1002 $0000_H$ | 1002 $0FFF_H$ | FLXA0 | √ | √ | √ | √ | √ | √ | √ | √ | ERRSLVFI |
| 1002 $1000_H$ | 1002 $1FFF_H$ | FLXA1 | — | — | — | √ | —*1 | √ | √ | √ | |
| 1002 $2000_H$ | 1002 $3FFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | ERRSLVHI |
| 1002 $4000_H$ | 1002 $4FFF_H$ | ETNA0 port0 | √ | √ | √ | √ | √ | √ | √ | √ | ERRSLVEI |
| 1002 $5000_H$ | 1002 $5FFF_H$ | ETNA0 port1 | — | — | — | √ | —*1 | √ | √ | √ | |
| 1002 $6000_H$ | 1002 $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | ERRSLVHI |
| 1003 $0000_H$ | 1003 $003F_H$ | MEMC (register) | — | — | — | √ | —*1 | √ | √ | √ | ERRSLVMI |
| 1003 $0000_H$ | 1003 $03FF_H$ | Unmapped area | — | — | — | — | — | — | — | — | |
| 1003 $0400_H$ | 1FFF $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | ERRSLVHI |
| 2000 $0000_H$ | 2000 $01FF_H$ | MEMC CS0 | — | — | — | √ | —*1 | √ | √ | √ | ERRSLVMI |
| 2000 $0200_H$ | 21FF $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | |
| 2200 $0000_H$ | 2200 $01FF_H$ | MEMC CS1 | — | — | — | √ | —*1 | √ | √ | √ | |
| 2200 $0200_H$ | 23FF $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | |
| 2400 $0000_H$ | 2400 $01FF_H$ | MEMC CS2 | — | — | — | √ | —*1 | √ | √ | √ | |
| 2400 $0200_H$ | 27FF $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | |
| 2800 $0000_H$ | 2800 $01FF_H$ | MEMC CS3 | — | — | — | √ | —*1 | √ | √ | √ | |
| 2800 $0200_H$ | 2BFF $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | |
| 2C00 $0000_H$ | F2FF $FFFF_H$ | Unmapped area | — | — | — | — | — | — | — | — | ERRSLVHI |

Note 1. The access does not output an error response.

### 4.3.7 Unmapped access error from H-bus master

The access from H-bus master is restricted to code flash, global RAM and local RAM area. Therefore when H-bus master accesses to other area, access error is signaled to ECM (No.79).

**Table 4.3 List of H-Bus master access**

| Start Address | End Address | Module Name | Unmapped Area Access Error Information |
|---|---|---|---|
| 0000 $0000_H$ | 0FFF $FFFF_H$ | Code Flash Area | |
| 1000 $0000_H$ | FE7F $FFFF_H$ | unmapped area | ERRSLVXI |
| FE80 $0000_H$ | FE9F $FFFF_H$ | PE2 Local RAM Area | |
| FEA0 $0000_H$ | FEBF $FFFF_H$ | PE1 Local RAM Area | |
| FEC0 $0000_H$ | FEDF $FFFF_H$ | unmapped area | ERRSLVXI |
| FEE0 $0000_H$ | FEFF $FFFF_H$ | Global RAM Area | |
| FE00 $0000_H$ | FFFF $FFFF_H$ | unmapped area | ERRSLVXI |

### 4.3.8 Debug area

For PE own debug area including self-area, an unmapped access error is not generated. For other PE's debug area, error response is returned to the access master. Debug area is the below

- Self: FA00 $0000_H$ to FAFF $FFFF_H$
- PE1: F900 $0000_H$ to F9FF $FFFF_H$
- PE2: F800 $0000_H$ to F8FF $FFFF_H$

### 4.3.9 Other area

When PE1, PE2 and DMA access to areas except for the above region, an error response is returned to the access master.

### 4.3.10 Registers

When an unmapped access to code flash, global RAM, P-bus and H-bus on-chip I/O area occurs, an error status is stored into register. In case of global RAM unmapped access error, see **Section 28.4.2, GRG**.

Table 4.4 List of Registers (1/2)

| Module Name | Register | Symbol | Address |
|---|---|---|---|
| CFGDSIC | Code Flash Guard Control Register (SIC) | MGDCFCTL_VCI2CFB | FFC4 8100$_H$ |
| CFGDSIC | Code Flash Guard Error Status Register (SIC) | MGDCFSTAT_VCI2CFB | FFC4 8104$_H$ |
| CFGDSIC | Code Flash Guard Error Access Type Register (SIC) | MGDCFTYPE_VCI2CFB | FFC4 810C$_H$ |
| CFGDPE1 | Code Flash Guard Control Register (PE1) | MGDCFCTL_PE1 | FFC4 8200$_H$ |
| CFGDPE1 | Code Flash Guard Error Status Register (PE1) | MGDCFSTAT_PE1 | FFC4 8204$_H$ |
| CFGDPE1 | Code Flash Guard Error Access Type Register (PE1) | MGDCFTYPE_PE1 | FFC4 820C$_H$ |
| CFGDPE2 | Code Flash Guard Control Register (PE2) | MGDCFCTL_PE2 | FFC4 8300$_H$ |
| CFGDPE2 | Code Flash Guard Error Status Register (PE2) | MGDCFSTAT_PE2 | FFC4 8304$_H$ |
| CFGDPE2 | Code Flash Guard Error Access Type Register (PE2) | MGDCFTYPE_PE2 | FFC4 830C$_H$ |
| ERRSLVFI | H-bus Unmapped Area Error Control Register (FLXA area) | ERRSLVFICTL | FFFA 0100$_H$ |
| ERRSLVFI | H-bus Unmapped Area Error Status Register (FLXA area) | ERRSLVFISTAT | FFFA 0104$_H$ |
| ERRSLVFI | H-bus Unmapped Area Error Access Type Register (FLXA area) | ERRSLVFITYPE | FFFA 010C$_H$ |
| ERRSLVEI | H-bus Unmapped Area Error Control Register (ETNA area) | ERRSLVEICTL | FFFA 0110$_H$ |
| ERRSLVEI | H-bus Unmapped Area Error Status Register (ETNA area) | ERRSLVEISTAT | FFFA 0114$_H$ |
| ERRSLVEI | H-bus Unmapped Area Error Access Type Register (ETNA area) | ERRSLVEITYPE | FFFA 011C$_H$ |
| ERRSLVMI | H-bus Unmapped Area Error Control Register (MEMC area) | ERRSLVMICTL | FFFA 0120$_H$ |
| ERRSLVMI | H-bus Unmapped Area Error Status Register (MEMC area) | ERRSLVMISTAT | FFFA 0124$_H$ |
| ERRSLVMI | H-bus Unmapped Area Error Access Type Register (MEMC area) | ERRSLVMITYPE | FFFA 012C$_H$ |
| ERRSLVHI | H-bus Unmapped Area Error Control Register (other H-bus on-chip I/O area) | ERRSLVHICTL | FFFA 0130$_H$ |
| ERRSLVHI | H-bus Unmapped Area Error Status Register (other H-bus on-chip I/O area) | ERRSLVHISTAT | FFFA 0134$_H$ |
| ERRSLVHI | H-bus Unmapped Area Error Access Type Register (other H-bus on-chip I/O area) | ERRSLVHITYPE | FFFA 013C$_H$ |
| ERRSLVXI | H-bus Unmapped Area Error Control Register (from H-bus master) | ERRSLVXICTL | FFFA 0140$_H$ |
| ERRSLVXI | H-bus Unmapped Area Error Status Register (from H-bus master) | ERRSLVXISTAT | FFFA 0144$_H$ |
| ERRSLVXI | H-bus Unmapped Area Error Access Type Register (from H-bus master) | ERRSLVXITYPE | FFFA 014C$_H$ |
| ERRSLV0AI | P-bus Unmapped Area Error Control Register (PFSS) | ERRSLVCTL_PBAREA | FFC4 C880$_H$ |
| ERRSLV0AI | P-bus Unmapped Area Error Status Register (PFSS) | ERRSLVSTAT_PBAREA | FFC4 C884$_H$ |
| ERRSLV0AI | P-bus Unmapped Area Error Access Type Register (PFSS) | ERRSLVTYPE_PBAREA | FFC4 C88C$_H$ |
| ERRSLV1AI | P-bus Unmapped Area Error Control Register (PBG1) | ERRSLV1AICTL | FFEE 0800$_H$ |
| ERRSLV1AI | P-bus Unmapped Area Error Status Register (PBG1) | ERRSLV1AISTAT | FFEE 0804$_H$ |
| ERRSLV1AI | P-bus Unmapped Area Error Access Type Register (PBG1) | ERRSLV1AITYPE | FFEE 080C$_H$ |
| ERRSLV2AI | P-bus Unmapped Area Error Control Register (PBG2) | ERRSLV2AICTL | FFDD D800$_H$ |
| ERRSLV2AI | P-bus Unmapped Area Error Status Register (PBG2) | ERRSLV2AISTAT | FFDD D804$_H$ |
| ERRSLV2AI | P-bus Unmapped Area Error Access Type Register (PBG2) | ERRSLV2AITYPE | FFDD D80C$_H$ |
| ERRSLV3AI | P-bus Unmapped Area Error Control Register (PBG3) | ERRSLV3AICTL | FFF9 4800$_H$ |
| ERRSLV3AI | P-bus Unmapped Area Error Status Register (PBG3) | ERRSLV3AISTAT | FFF9 4804$_H$ |
| ERRSLV3AI | P-bus Unmapped Area Error Access Type Register (PBG3) | ERRSLV3AITYPE | FFF9 480C$_H$ |

Table 4.4 List of Registers (2/2)

| Module Name | Register | Symbol | Address |
|---|---|---|---|
| ERRSLV4AI | P-bus Unmapped Area Error Control Register (PBG4) | ERRSLV4AICTL | FFF9 0800$_H$ |
| ERRSLV4AI | P-bus Unmapped Area Error Status Register (PBG4) | ERRSLV4AISTAT | FFF9 0804$_H$ |
| ERRSLV4AI | P-bus Unmapped Area Error Access Type Register (PBG4) | ERRSLV4AITYPE | FFF9 080C$_H$ |

**NOTE**

The registers with symbols "_VCI2CFB", "_PE1", and "_PE2" as suffixes are provided to the particular code flash guard registers: the registers with "_VCI2CFB" are provided for access from the system interconnect to code flash, the registers with "_PE1" are provided for access from the CPU1 to code flash, and the registers with "_PE2" are provided for access from the CPU2 to code flash.

#### 4.3.10.1 MGDCFCTL_VCI2CFB/PE1/PE2 — Code Flash Guard Control Register (VCI2CFB/PE1/PE2)

See **(2) (a) FLI Guard Control Register (VCI2CFB, PE1, PE2) (MGDCFCTL_{VCI2CFB, PE1, PE2})** in FLI (Code-Flash) Guard in **Section 3.2.4.4**.

#### 4.3.10.2 MGDCFSTAT_VCI2CFB/PE1/PE2 — Code Flash Guard Error Status Register (VCI2CFB/PE1/PE2)

See **(2) (b) FLI Guard Error Status Register (VCI2CFB, PE1, PE2) (MGDCFSTAT_{VCI2CFB, PE1, PE2})** in FLI (Code-Flash) Guard in **Section 3.2.4.4**.

#### 4.3.10.3 MGDCFTYPE_VCI2CFB/PE1/PE2_ — Code Flash Guard Error Access Type Register (VCI2CFG/PE1/PE2)

See **(2) (c) FLI Guard Error Access Type Register (VCI2CFB, PE1, PE2) (MGDCFTYPE_{VCI2CFB, PE1, PE2})** in FLI (Code-Flash) Guard in **Section 3.2.4.4**.

#### 4.3.10.4 H-bus Unmapped area access error registers

The functionality is the same as ERRSLV for H-bus guard. See (4), ERRSLVxxCTL, (5), ERRSLVxxSTAT and (6), ERRSLVxxTYPE in **Section 28.4.4.1**.

#### 4.3.10.5 P-bus Unmapped area access error registers for PFSS

See (2) (a) ERRSLV Control Register for PFSS APB Area Error (ERRSLVCTL_PBAREA), (b) ERRSLV Status Register for PFSS APB Area Error (ERRSLVSTAT_PBAREA), (d) ERRSLV Error Transfer Type Register for PFSS APB Area Error (ERRSLVTYPE_PBAREA), in APB Un-implemented Area Error in **Section 3.2.4.5**.

#### 4.3.10.6 P-bus Unmapped area access error registers

The functionality is the same as ERRSLV for P-bus guard. See (11), ERRSLVxCTL - ERRSLV Control Register for P-bus Guard, (12), ERRSLVxSTAT - ERRSLV Status Register for P-Bus Guard and (13), ERRSLVxTYPE - ERRSLV Error Transfer Type Register for P-Bus Guard in **Section 28.4.3.2**.

## 4.4 Guard function

The following table shows the guard function for each of memory area. In additional to these guard function, each of PE has MPU function in itself.In detail see **28.4.1**

| Memory Area | Access from | Guard function |
|---|---|---|
| P-Bus Area | All bus masters | P-Bus Guard (PBG) |
| LPB Area | own PE | Internal Peripheral Guard (IPG) |
| | other bus masters | PE Guard (PEG) |
| Data Flash Area | All bus masters | Data Flash Guard (DFG) |
| Global RAM Area | All bus masters | GRAM Guard (GRG) |
| Local RAM Area | All bus masters (except own PE) | PE Guard (PEG) |
| H-Bus Area | All bus masters | H-Bus Guard (HBG) |
| Code Flash Area | All bus masters (except ICUMC) | Code Flash Guard (CFG) |

## 4.5 Difference among P1M-C, P1H-C and P1H-CE

See **Figure 4.1**.

# Section 5 Operating Modes

## 5.1 Features

The device supports 3 User modes. Operating modes are selected by the mode terminals FLMD0, FLMD1, MODE0 and MODE1. **Table 5.1** shows the list of the operating modes. Device Mode decoding is only executed after Terminal reset.

**Table 5.1 Mode List**

| | mode pin | | | | |
|---|---|---|---|---|---|
| | **FLMD0** | **FLMD1** | **MODE0** | **MODE1** | **mode** |
| User mode | 0 | x | x | x | Normal Operation Mode |
| | 1 | 0 | x | x | Serial Flash Programming Mode<br>(Following serial I/F are supported<br>3-wire Clocked serial Interface,<br>2-wire UART,<br>1-wire UART) |
| | 1 | 1 | 0 | 1 | Boundary SCAN Mode |

### 5.1.1 Normal Operation Mode

This mode is the default mode for execution of application software. Also self-tests and the On-chip Debug capabilities use this mode. Wake up condition for calibration is generated from external via debug interface. After reset release, instruction fetch is carried out from the user mat. For reset vector of each CPU, see **Section 32, Flash Memory**.

### 5.1.2 Serial Flash Programming Mode

This mode is initiated by an external programming device. Use it to store application code or application data into the flash memories of the device. After reset release, the MCU boots up from the on-chip boot program and starts connection in the specified transmission method. For details, see **Section 32, Flash Memory**.

### 5.1.3 Boundary SCAN Mode

This mode allows Boundary SCAN tests compliant with IEEE Standard 1149.1. In this mode, CPU is in reset status. For details, see **Section 34, Boundary Scan**.

## 5.2 Input Pins

**Table 5.2** shows the pin information for mode setting. All mode terminals are taken in Mode Register. FLMD1 and MODE [1:0] can be used as GPIO. The level on the FLMD0 pin can be set to prohibit programming and erasure of the code flash memory. When the FLMD0 pin is at the high level, programming and erasure of the memory are permitted; when it is at the low level it is prohibited to program or erase the memory. For details, see **Section 32, Flash Memory**.

**Table 5.2 Pin information for mode setting**

| Pin Name | I/O | Function |
|---|---|---|
| FLMD0 | input | Operating mode select pin |
| FLMD1 | input | Operating mode select pin |
| MODE0 | input | Operating mode select pin |
| MODE1 | input | Operating mode select pin |

## 5.3 Register Description

### 5.3.1 List of Registers

Table 5.3 Register Configuration

| Address | Register Name | Description | Access Width | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | other |
| FFF8 0104 | MODE | Mode Register | 32 | 0000 000X$_H$ | PBG4#0. PG4-SC3 | — |

Table 5.4 Register Reset condition

| Register Name | Reset condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| MODE | √ | √*1 | | | |

Note 1. Terminal Reset only

The status of the mode pin is taken in this register when Terminal reset is released.

### 5.3.2 MODE — Mode Register

This register indicates the Operating Mode of the device.

The status of the mode pin is taken in this register when Terminal reset is released.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 0104$_H$

**Value after reset:** 0000 000X$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MODE1 | MODE0 | FLMD1 | FLMD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | X |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 5.5 MODE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. |
| 3 | MODE1 | Mode 1<br>This bit indicates the Status of the Latch for Mode Pin MODE 1<br>0: Low-level detection<br>1: High-level detection |
| 2 | MODE0 | Mode 0<br>This bit indicates the Status of the Latch for Mode Pin MODE 0<br>0: Low-level detection<br>1: High-level detection |
| 1 | FLMD1 | FLMD 1<br>This bit indicates the Status of the Latch for Mode Pin FLMD 1<br>0: Low-level detection<br>1: High-level detection |
| 0 | FLMD0 | FLMD 0<br>This bit indicates the Status of the Latch for Mode Pin FLMD 0<br>0: Low-level detection<br>1: High-level detection |

# Section 6 Interrupt Controller (INTC)

## 6.1 Features

The act of branching from a currently running program to a different program in response to an event is called an exception. This microcontroller supports the following types of exceptions.

The details on exceptions are described in *the RH850G3M User's Manual: Software*.

**Table 6.1 List of Exception Sources**

| Name | Symbol | Source | Priority | Saved to |
|---|---|---|---|---|
| Reset | RESET | Reset input | High | — |
| Debug exception (asynchronous) | AsyncDB (NMI) | Asynchronous debug event | | DB |
| FE level non-maskable interrupt*1 | FENMI (NMI) | FENMI input | | FE |
| System error exception | SYSERR (NMI) | SYSERR input | | FE |
| FE level maskable interrupt*1 | FEINT (NMI) | FEINT input | | FE |
| FPU exception | FPU(NMI) | Execution of an FPU instruction | | EI |
| EI level maskable interrupt*1 | INT | Interrupt controller | | EI |
| Debug exception (synchronous) | SyncDB | Synchronous debug event | | DB |
| Memory protection violation (execution right) | MIP | Memory protection violation | | FE |
| System error exception | SYSERR | Error input at instruction fetch | | FE |
| Reserved instruction exception | RIE | Execution of reserved instruction | | FE |
| Coprocessor unusable exception | UCPOP | Execution of coprocessor instruction | | FE |
| Privileged instruction exception | PIE | Execution of reserved instruction | | FE |
| Misalign exception | MAE | Generation of misalign access | | FE |
| Memory protection exception (access right) | MDP | Memory protection violation | | FE |
| Debug trap | DBTRAP | Execution of DBTRAP instruction | | DB |
| FPU exception (precise) | FPP | Execution of an FPU instruction | | EI |
| Runtime monitor trap | RMTRAP | Execution of RMTRAP instruction | | DB |
| System call | SYSCALL | Execution of SYSCALL instruction | | EI |
| FE level trap | FETRAP | Execution of FETRAP instruction | | FE |
| EI level trap 0 | EITRAP0 | Execution of TRAP0n instruction | | EI |
| EI level trap 1 | EITRAP1 | Execution of TRAP1n instruction | Low | EI |

Note 1. The description of these exceptions and interrupts are subject to this section.

### (1) Interrupts

The following three exceptions from **Table 6.1** are called interrupts, and are thus described in this section.

- FE level non-maskable interrupt (FENMI)
  An FENMI interrupt is acknowledged even if another FE level interrupt – FENMI or FEINT – is in service.
  - FENMI is acknowledged even if the CPU system register PSW.NP = 1.
  - Return from FENMI interrupt is not possible, recover is disabled
- FE level maskable interrupt (FEINT)
  An FEINT interrupt can be acknowledged if another FE level interrupt – FENMI or FEINT – is not in service.
  - FEINT can be acknowledged if the CPU system register PSW.NP = 0. It is masked if PSW.NP = 1.
  - Return enabled, recover enabled
  - Highest priority interrupt (except FENMI)
- EI level maskable interrupt (EIINT)
  An INT interrupt can be acknowledged if FE level interrupt – FENMI or FEINT – is not in service.
  - EIINT can be acknowledged if the CPU system register PSW.NP = 0.
    It is masked if PSW.NP = 1, EIINT with higher priority is being processed, or PSW.ID = 1.
  - Return enabled, recover enabled.
  - Interrupt masking can be specified per interrupt channel.
  - 16 interrupt priority levels can be specified for each interrupt channel
  - In this section, the EIINT that corresponds to interrupt channel n is indicated by “EIINTn”, whereas the EIINT that corresponds to interrupt source xxxis indicated by “INTxxx”.
  - Interrupts from peripheral Modules can be used for additional trigger events like DMA.

For the PSW register, see **Section 3.2.1.2(2)(e), PSW — Program status word** and *RH850G3M User's Manual: Software*.

RH850 core provide the DI (disable interrupts) instruction to disable all EI level maskable interrupts until an EI (enable interrupts) instruction is issued.

Exceptions, which are local to a core, are raised locally. Exceptions from a global source, e.g. ECC failures on common RAM areas or peripheral areas (e.g. CAN RAM, Flexray RAM and so on) are commonly reported to all cores. All CPUs of a multicore device have the means for each CPU core to issue maskable interrupts to each other core.

**NOTE**

Return: Indicates whether execution restart from the last position at which program execution was interrupted is possible.

Recover: Indicates whether recovery to the processor status (status of processor resources including general-purpose registers and system registers) at the time of program execution interruption is possible.

These interrupt sources are described in the following pages.

Table 6.2 Clock Supply

| Unit Name | Clock for the Unit | Supply Clock Name |
|---|---|---|
| INTC1 | Operation clock | CLK_CPU |
| INTC2 | Operation clock | CLK_HSB |

## 6.2 RH850/P1x-C Interrupt sources

### 6.2.1 RH850/P1x-C Interrupt sources

#### (1) FE level non-maskable interrupts

(a) Priority group

See **Table 6.1, List of Exception Sources**.

(b) Return PC

An FE non-maskable interrupt does not allow to return or recover.

(c) Status register

See **Section 6.4.3, FNC — FE level NMI status register**.

(d) Return instruction

None

Table 6.3 FE level non-maskable interrupt requests

| Interrupt | | | Interrupt request | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Control register | | | | | | | |
| Symbol | Name | Address | Name | Cause | Unit | Priority group | Exception code | Handler address offset |
| FENMI | FNC | FFFE EA78$_H$ | SWDTA0NMI | Secure WDT overflow interrupt | WDT | *1 | E0$_H$ | 0E0$_H$ |

Note 1. See **Table 6.1, List of Exception Sources**.

#### (2) FE level maskable interrupts

There are three sources for the FEINT: ECM and STM interrupt and terminal NMI.

These three sources are mapped to the interrupt controller INTC1. There is a status flag register and an event clear register for each source. The event status flag has to be cleared by software.

The event status flag should be cleared by software in the beginning of the interrupt service routine. If there is a pending interrupt flag after leaving the interrupt service routine the interrupt will be asserted again until all flags in FEINTESTAT are cleared.

(a) Priority group

See **Table 6.1, List of Exception Sources**.

(b) Return PC

The PC return from an interrupt handling routine by the FERET instruction is the suspended PC (current PC).

(c) Status register

See **Section 6.4.4, FIC — FE level maskable interrupt status register**.

(d) Return instruction

FERET

**Table 6.4 FE level maskable interrupt requests**

| Interrupt | | | Interrupt request | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Control register | | | | | | | |
| Symbol | Name | Address | Name | Cause | Unit | Priority group | Exception code | Handler address offset |
| FEINT for PE1 | FIC | FFFE EA7A$_H$ | — | NMI pin | Port | *1 | 0F0$_H$ (PE1) | 0F0$_H$ (PE1) |
| | | | — | non mask-able interrupt (STM interrupt 8 for STM0) , | STM | | | |
| | | | — | non mask-able interrupt from ECM | ECM | | | |
| FEINT for PE2 | FIC | FFFE EA7A$_H$ | — | NMI pin | Port | *1 | 0F0$_H$ (PE2) | 0F0$_H$ (PE2) |
| | | | — | non mask-able interrupt (STM interrupt 8 for STM1) , | STM | | | |
| | | | — | non mask-able interrupt from ECM | ECM | | | |

Note 1. See **Table 6.1, List of Exception Sources**.

The source of the FEINT interrupt can be evaluated by a dedicated flag register. See **Section 6.4.5, FEINTFn — FE level maskable interrupt event status register** and **Section 6.8.7, FEINT Source selection** for details.

### (3) EI level maskable interrupts

(a) Priority group

See **Table 6.1, List of Exception Sources**.

(b) Return PC

The PC return from an interrupt handling routine by the EIRET instruction is the suspended PC (current PC).

(c) Control register

EI level maskable interrupt control register

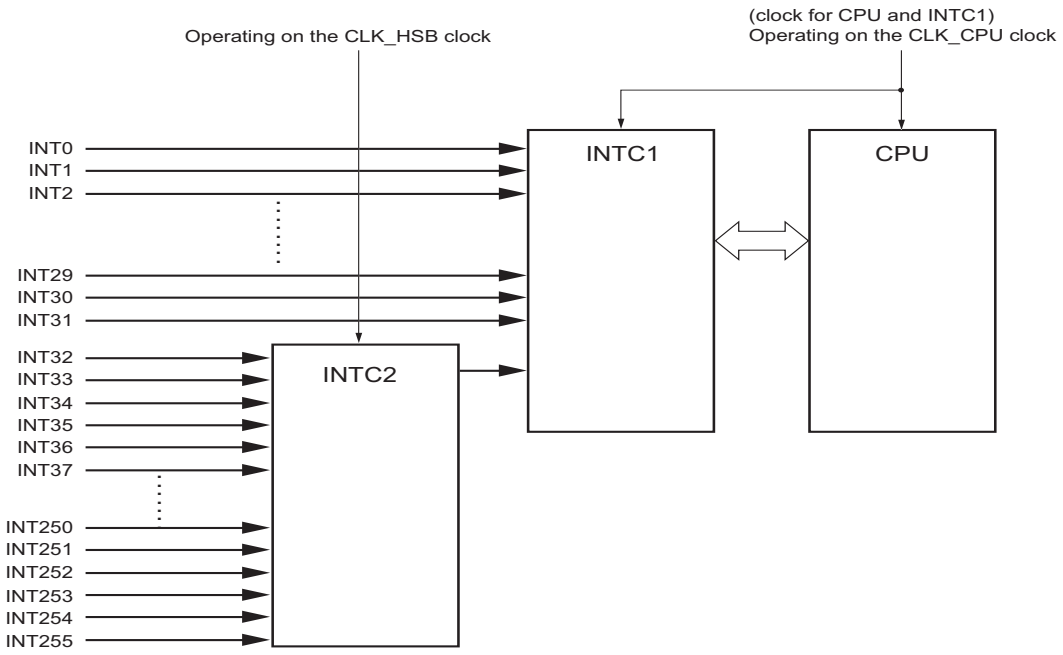
See **Section 6.4.1, EICn — EI level interrupt control registers**.

(d) Return instruction

EIRET

(e) Configuration

EI level maskable interrupt is supported total 256 channels with cascade connection of INTC1 and INTC2.




**Note:** In case of multicore configuration (P1H-C(4MB), P1H-C(8MB), and P1H-CE) an INTC1 is implmented in each PE. INTC2 is shared among the PEs.

**Figure 6.1 Configuration Diagram of EI-level Maskable Interrupt**

**CAUTION**

**As CLK_HSB is the operating clock for INTC2, INT32 to INT255 interrupts, which are connected to INTC2, are delayed more than the interrupts directly connected to INTC1.**

**Table 6.5** lists EI-level maskable interrupts.

0 to 31: Connected to INTC1. 32 to 255: Connected to INTC2

**Table 6.5 RH850/P1x-C EI level maskable interrupt sources (1/7)**

| Interrupt | | | Interrupt request | | | Detection type*2 | Exception code | Handler address (offset)*6 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Control register | | | | | | | Direct jumping to an address | | |
| Channel*1 | Name | Address | Name | Cause | Unit | | | RINT = 0*3 | RINT = 1*4 | Reference to a table*5 |
| 0 | EIC0 | FFFE EA00$_H$ | INTECM0MI/ INTECM1MI | mask-able interrupt from ECM | ECM | Edge | 1000$_H$ | Offset address does not differ from channel to channel; it is determined according to the priority order, and is taken from an address between +100$_H$ to +1F0$_H$. | Offset address is uniformly +100$_H$ irrespective of the priority order. | +000$_H$ |
| 1 | EIC1 | FFFE EA02$_H$ | INTIPIR0 | IPIR_CH1 (between PE interrupt) | IPIR | Edge | 1001$_H$ | | | +004$_H$ |
| 2 | EIC2 | FFFE EA04$_H$ | INTIPIR1 | IPIR_CH2 (between PE interrupt) | IPIR | Edge | 1002$_H$ | | | +008$_H$ |
| 3 | EIC3 | FFFE EA06$_H$ | INTICUP1/INTICUP2 | interrupt from ICUMC | ICUMC | Edge | 1003$_H$ | | | +00C$_H$ |
| 4 | EIC4 | FFFE EA08$_H$ | INTBN0 | broadcast notification 0 | INTC2 | Edge | 1004$_H$ | | | +010$_H$ |
| 5 | EIC5 | FFFE EA0A$_H$ | INTBN1 | broadcast notification 1 | INTC2 | Edge | 1005$_H$ | | | +014$_H$ |
| 6 | EIC6 | FFFE EA0C$_H$ | INTBN2 | broadcast notification 2 | INTC2 | Edge | 1006$_H$ | | | +018$_H$ |
| 7 | EIC7 | FFFE EA0E$_H$ | INTBN3 | broadcast notification 3 | INTC2 | Edge | 1007$_H$ | | | +01C$_H$ |
| 8 | EIC8 | FFFE EA10$_H$ | INTWDTA0/ INTWDTA1 | Interval timer interrupt(75% interrupt) | WDTA | Edge | 1008$_H$ | | | +020$_H$ |
| 9 | EIC9 | FFFE EA12$_H$ | INTSW0 | Software interrupt0 | SW | Level | 1009$_H$ | | | +024$_H$ |
| 10 | EIC10 | FFFE EA14$_H$ | INTSW1 | Software interrupt1 | SW | Level | 100A$_H$ | | | +028$_H$ |
| 11 | EIC11 | FFFE EA16$_H$ | INTSW2 | Software interrupt2 | SW | Level | 100B$_H$ | | | +02C$_H$ |
| 12 | EIC12 | FFFE EA18$_H$ | INTSTM02 | STM interrupt 2 (STM0) | STM | Edge | 100C$_H$ | | | +030$_H$ |
| 13 | EIC13 | FFFE EA1A$_H$ | INTSTM03 | STM interrupt 3 (STM0) | STM | Edge | 100D$_H$ | | | +034$_H$ |
| 14 | EIC14 | FFFE EA1C$_H$ | INTSTM04 | STM interrupt 4 (STM0) | STM | Edge | 100E$_H$ | | | +038$_H$ |
| 15 | EIC15 | FFFE EA1E$_H$ | INTSTM05 | STM interrupt 5 (STM0) | STM | Edge | 100F$_H$ | | | +03C$_H$ |
| 16 | EIC16 | FFFE EA20$_H$ | INTDMA0 | DMA transfer completion for ch0 / DMA count match interrupt for ch0 | DMA | Edge | 1010$_H$ | | | +040$_H$ |
| 17 | EIC17 | FFFE EA22$_H$ | INTDMA1 | DMA transfer completion for ch1 / DMA count match interrupt for ch1 | DMA | Edge | 1011$_H$ | | | +044$_H$ |
| 18 | EIC18 | FFFE EA24$_H$ | INTDMA2 | DMA transfer completion for ch2 / DMA count match interrupt for ch2 | DMA | Edge | 1012$_H$ | | | +048$_H$ |
| 19 | EIC19 | FFFE EA26$_H$ | INTDMA3 | DMA transfer completion for ch3 / DMA count match interrupt for ch3 | DMA | Edge | 1013$_H$ | | | +04C$_H$ |
| 20 | EIC20 | FFFE EA28$_H$ | INTDMA4 | DMA transfer completion for ch4 / DMA count match interrupt for ch4 | DMA | Edge | 1014$_H$ | | | +050$_H$ |
| 21 | EIC21 | FFFE EA2A$_H$ | INTDMA5 | DMA transfer completion for ch5 / DMA count match interrupt for ch5 | DMA | Edge | 1015$_H$ | | | +054$_H$ |
| 22 | EIC22 | FFFE EA2C$_H$ | INTDMA6 | DMA transfer completion for ch6 / DMA count match interrupt for ch6 | DMA | Edge | 1016$_H$ | | | +058$_H$ |
| 23 | EIC23 | FFFE EA2E$_H$ | INTDMA7 | DMA transfer completion for ch7 / DMA count match interrupt for ch7 | DMA | Edge | 1017$_H$ | | | +05C$_H$ |
| 24 | EIC24 | FFFE EA30$_H$ | INTDMA8 | DMA transfer completion for ch8 / DMA count match interrupt for ch8 | DMA | Edge | 1018$_H$ | | | +060$_H$ |
| 25 | EIC25 | FFFE EA32$_H$ | INTDMA9 | DMA transfer completion for ch9 / DMA count match interrupt for ch9 | DMA | Edge | 1019$_H$ | | | +064$_H$ |
| 26 | EIC26 | FFFE EA34$_H$ | INTDMA10 | DMA transfer completion for ch10 / DMA count match interrupt for ch10 | DMA | Edge | 101A$_H$ | | | +068$_H$ |
| 27 | EIC27 | FFFE EA36$_H$ | INTDMA11 | DMA transfer completion for ch11 / DMA count match interrupt for ch11 | DMA | Edge | 101B$_H$ | | | +06C$_H$ |
| 28 | EIC28 | FFFEEA38$_H$ | INTDMA12 | DMA transfer completion for ch12 / DMA count match interrupt for ch12 | DMA | Edge | 101C$_H$ | | | +070$_H$ |
| 29 | EIC29 | FFFE EA3A$_H$ | INTDMA13 | DMA transfer completion for ch13 / DMA count match interrupt for ch13 | DMA | Edge | 101D$_H$ | | | +074$_H$ |
| 30 | EIC30 | FFFE EA3C$_H$ | INTDMA14 | DMA transfer completion for ch14 / DMA count match interrupt for ch14 | DMA | Edge | 101E$_H$ | | | +078$_H$ |
| 31 | EIC31 | FFFE EA3E$_H$ | INTDMA15 | DMA transfer completion for ch15 / DMA count match interrupt for ch15 | DMA | Edge | 101F$_H$ | | | +07C$_H$ |

Table 6.5 RH850/P1x-C EI level maskable interrupt sources (2/7)

| Interrupt | | | Interrupt request | | | Detection type*2 | Exception code | Handler address (offset)*6 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Control register | | | | | | | Direct jumping to an address | | |
| Channel*1 | Name | Address | Name | Cause | Unit | | | RINT = 0*3 | RINT = 1*4 | Reference to a table*5 |
| 32 | EIC32 | FFFF B040$_H$ | INTP0 | External Interrupt0 | Pin | Edge | 1020$_H$ | Offset address does not differ from channel to channel; it is determined according to the priority order, and is taken from an address between +100$_H$ to +1F0$_H$. | Offset address is uniformly +100$_H$ irrespective of the priority order. | +080$_H$ |
| 33 | EIC33 | FFFF B042$_H$ | INTP1 | External Interrupt1 | Pin | Edge | 1021$_H$ | | | +084$_H$ |
| 34 | EIC34 | FFFF B044$_H$ | INTP2 | External Interrupt2 | Pin | Edge | 1022$_H$ | | | +088$_H$ |
| 35 | EIC35 | FFFF B046$_H$ | INTP3 | External Interrupt3 | Pin | Edge | 1023$_H$ | | | +08C$_H$ |
| 36 | EIC36 | FFFF B048$_H$ | INTP4 | External Interrupt4 | Pin | Edge | 1024$_H$ | | | +090$_H$ |
| 37 | EIC37 | FFFF B04A$_H$ | INTOTS0OTE | Temperature sensor error interrupt | TEMPS | Edge | 1025$_H$ | | | +094$_H$ |
| 38 | EIC38 | FFFF B04C$_H$ | INTOTS0OTI | temperature measurement end interrupt | TEMPS | Edge | 1026$_H$ | | | +098$_H$ |
| 39 | EIC39 | FFFF B04E$_H$ | INTOTS0OTULI | Triggered if state machine change the stage by the temperature rising or falling in the guaranteed temperature range | TEMPS | Edge | 1027$_H$ | | | +09C$_H$ |
| 40 | EIC40 | FFFF B050$_H$ | INTFLXNTU | NTU loss interrupt | NTU | Level | 1028$_H$ | | | +0A0$_H$ |
| 41 | EIC41 | FFFF B052$_H$ | INTSTM00 | STM interrupt 0 (STM0) | STM | Edge | 1029$_H$ | | | +0A4$_H$ |
| 42 | EIC42 | FFFF B054$_H$ | INTSTM01 | STM interrupt 1 (STM0) | STM | Edge | 102A$_H$ | | | +0A8$_H$ |
| 43 | EIC43 | FFFF B056$_H$ | INTSTM10 | STM interrupt 0 (STM1) | STM | Edge | 102B$_H$ | | | +0AC$_H$ |
| 44 | EIC44 | FFFF B058$_H$ | INTSTM11 | STM interrupt 1 (STM1) | STM | Edge | 102C$_H$ | | | +0B0$_H$ |
| 45 | EIC45 | FFFF B05A$_H$ | INTSTM12 | STM interrupt 2 (STM1) | STM | Edge | 102D$_H$ | | | +0B4$_H$ |
| 46 | EIC46 | FFFF B05C$_H$ | INTSTM13 | STM interrupt 3 (STM1) | STM | Edge | 102E$_H$ | | | +0B8$_H$ |
| 47 | EIC47 | FFFF B05E$_H$ | INTGTM0ERR | GTM Error Interrupt | GTM | Edge | 102F$_H$ | | | +0BC$_H$ |
| 48 | EIC48 | FFFF B060$_H$ | INTGTM0AEI | AEI Shared interrupt | GTM | Edge | 1030$_H$ | | | +0C0$_H$ |
| 49 | EIC49 | FFFF B062$_H$ | INTGTM0ARU0 | ARU_NEW_DATA0 interrupt | GTM | Edge | 1031$_H$ | | | +0C4$_H$ |
| 50 | EIC50 | FFFF B064$_H$ | INTGTM0ARU1 | ARU_NEW_DATA1 interrupt | GTM | Edge | 1032$_H$ | | | +0C8$_H$ |
| 51 | EIC51 | FFFF B066$_H$ | INTGTM0ARU2 | ARU_ACC_ACK interrupt | GTM | Edge | 1033$_H$ | | | +0CC$_H$ |
| 52 | EIC52 | FFFF B068$_H$ | INTGTM0CMP | CMP Shared interrupt | GTM | Edge | 1034$_H$ | | | +0D0$_H$ |
| 53 | EIC53 | FFFF B06A$_H$ | INTGTM0TIM00 | TIM Shared interrupts (TIM0_IRQ0) | GTM | Edge | 1035$_H$ | | | +0D4$_H$ |
| 54 | EIC54 | FFFF B06C$_H$ | INTGTM0TIM02 | TIM Shared interrupts (TIM0_IRQ2) | GTM | Edge | 1036$_H$ | | | +0D8$_H$ |
| 55 | EIC55 | FFFF B06E$_H$ | INTGTM0TIM04 | TIM Shared interrupts (TIM0_IRQ4) | GTM | Edge | 1037$_H$ | | | +0DC$_H$ |
| 56 | EIC56 | FFFF B070$_H$ | INTGTM0TIM06 | TIM Shared interrupts (TIM0_IRQ6) | GTM | Edge | 1038$_H$ | | | +0E0$_H$ |
| 57 | EIC57 | FFFF B072$_H$ | INTGTM0TIM10 | TIM Shared interrupts (TIM1_IRQ0) | GTM | Edge | 1039$_H$ | | | +0E4$_H$ |
| 58 | EIC58 | FFFF B074$_H$ | INTGTM0TIM12 | TIM Shared interrupts (TIM1_IRQ2) | GTM | Edge | 103A$_H$ | | | +0E8$_H$ |
| 59 | EIC59 | FFFF B076$_H$ | INTGTM0TIM14 | TIM Shared interrupts (TIM1_IRQ4) | GTM | Edge | 103B$_H$ | | | +0EC$_H$ |
| 60 | EIC60 | FFFF B078$_H$ | INTGTM0TIM16 | TIM Shared interrupts (TIM1_IRQ6) | GTM | Edge | 103C$_H$ | | | +0F0$_H$ |
| 61 | EIC61 | FFFF B07A$_H$ | INTGTM0MCS00 | MCS Interrupt for channel (MCS0_IRQ0) | GTM | Edge | 103D$_H$ | | | +0F4$_H$ |
| 62 | EIC62 | FFFF B07C$_H$ | INTGTM0MCS02 | MCS Interrupt for channel (MCS0_IRQ2) | GTM | Edge | 103E$_H$ | | | +0F8$_H$ |
| 63 | EIC63 | FFFF B07E$_H$ | INTGTM0MCS04 | MCS Interrupt for channel (MCS0_IRQ4) | GTM | Edge | 103F$_H$ | | | +0FC$_H$ |
| 64 | EIC64 | FFFF B080$_H$ | INTGTM0MCS06 | MCS Interrupt for channel (MCS0_IRQ6) | GTM | Edge | 1040$_H$ | | | +100$_H$ |
| 65 | EIC65 | FFFF B082$_H$ | INTGTM0MCS08 | MCS Interrupt for channel (MCS0_IRQ8) | GTM | Edge | 1041$_H$ | | | +104$_H$ |
| 66 | EIC66 | FFFF B084$_H$ | INTGTM0MCS11 | MCS Interrupt for channel (MCS1_IRQ1) | GTM | Edge | 1042$_H$ | | | +108$_H$ |
| 67 | EIC67 | FFFF B086$_H$ | INTGTM0MCS13 | MCS Interrupt for channel (MCS1_IRQ3) | GTM | Edge | 1043$_H$ | | | +10C$_H$ |
| 68 | EIC68 | FFFF B088$_H$ | INTGTM0MCS15 | MCS Interrupt for channel (MCS1_IRQ5) | GTM | Edge | 1044$_H$ | | | +110$_H$ |
| 69 | EIC69 | FFFF B08A$_H$ | INTGTM0ATOM00 | ATOM Shared interrupts (ATOM0_IRQ0) | GTM | Edge | 1045$_H$ | | | +114$_H$ |
| 70 | EIC70 | FFFF B08C$_H$ | INTGTM0ATOM02 | ATOM Shared interrupts (ATOM0_IRQ2) | GTM | Edge | 1046$_H$ | | | +118$_H$ |
| 71 | EIC71 | FFFF B08E$_H$ | INTGTM0ATOM04 | ATOM Shared interrupts (ATOM0_IRQ4) | GTM | Edge | 1047$_H$ | | | +11C$_H$ |
| 72 | EIC72 | FFFF B090$_H$ | INTGTM0ATOM06 | ATOM Shared interrupts (ATOM0_IRQ6) | GTM | Edge | 1048$_H$ | | | +120$_H$ |
| 73 | EIC73 | FFFF B092$_H$ | INTGTM0ATOM10 | ATOM Shared interrupts (ATOM1_IRQ0) | GTM | Edge | 1049$_H$ | | | +124$_H$ |
| 74 | EIC74 | FFFF B094$_H$ | INTGTM0ATOM12 | ATOM Shared interrupts (ATOM1_IRQ2) | GTM | Edge | 104A$_H$ | | | +128$_H$ |
| 75 | EIC75 | FFFF B096$_H$ | INTGTM0ATOM14 | ATOM Shared interrupts (ATOM1_IRQ4) | GTM | Edge | 104B$_H$ | | | +12C$_H$ |

Table 6.5 RH850/P1x-C EI level maskable interrupt sources (3/7)

| Interrupt | | | Interrupt request | | | Detection type*2 | Exception code | Handler address (offset)*6 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Control register | | | | | | | Direct jumping to an address | | |
| Channel*1 | Name | Address | Name | Cause | Unit | | | RINT = 0*3 | RINT = 1*4 | Reference to a table*5 |
| 76 | EIC76 | FFFF B098$_H$ | INTGTM0ATOM16 | ATOM Shared interrupts (ATOM1_IRQ6) | GTM | Edge | 104C$_H$ | Offset address does not differ from channel to channel; it is determined according to the priority order, and is taken from an address between +100$_H$ to +1F0$_H$. | Offset address is uniformly +100$_H$ irrespective of the priority order. | +130$_H$ |
| 77 | EIC77 | FFFF B09A$_H$ | INTGTM0ATOM20 | ATOM Shared interrupts (ATOM2_IRQ0) | GTM | Edge | 104D$_H$ | | | +134$_H$ |
| 78 | EIC78 | FFFF B09C$_H$ | INTGTM0ATOM22 | ATOM Shared interrupts (ATOM2_IRQ2) | GTM | Edge | 104E$_H$ | | | +138$_H$ |
| 79 | EIC79 | FFFF B09E$_H$ | INTGTM0ATOM24 | ATOM Shared interrupts (ATOM2_IRQ4) | GTM | Edge | 104F$_H$ | | | +13C$_H$ |
| 80 | EIC80 | FFFF B0A0$_H$ | INTADCF0ERR | Error Interrupt for ADCF0 | ADC | Edge | 1050$_H$ | | | +140$_H$ |
| 81 | EIC81 | FFFF B0A2$_H$ | INTADCF0I0 | Trigger group for SG0 of ADCF0 | ADC | Edge | 1051$_H$ | | | +144$_H$ |
| 82 | EIC82 | FFFF B0A4$_H$ | INTADCF0I1 | Trigger group for SG1 of ADCF0 | ADC | Edge | 1052$_H$ | | | +148$_H$ |
| 83 | EIC83 | FFFF B0A6$_H$ | INTADCF0I2 | Trigger group for SG2 of ADCF0 | ADC | Edge | 1053$_H$ | | | +14C$_H$ |
| 84 | EIC84 | FFFF B0A8$_H$ | INTADCF0I3 | Trigger group for SG3 of ADCF0 | ADC | Edge | 1054$_H$ | | | +150$_H$ |
| 85 | EIC85 | FFFF B0AA$_H$ | INTADCF0I4 | Trigger group for SG4 of ADCF0 | ADC | Edge | 1055$_H$ | | | +154$_H$ |
| 86 | EIC86 | FFFF B0AC$_H$ | INTCSIH0TIRE | Transmission Interrupt Reception Error signal for CSIH_0 | CSIH | Edge | 1056$_H$ | | | +158$_H$ |
| 87 | EIC87 | FFFF B0AE$_H$ | INTCSIH0TIR | Transmission Interrupt Reception signal for CSIH_0 | CSIH | Edge | 1057$_H$ | | | +15C$_H$ |
| 88 | EIC88 | FFFF B0B0$_H$ | INTCSIH0TIC | Transmission Interrupt Communication signal for CSIH_0 | CSIH | Edge | 1058$_H$ | | | +160$_H$ |
| 89 | EIC89 | FFFF B0B2$_H$ | INTCSIH0TIJC | Transmission Interrupt for JOB Completion for CSIH_0 | CSIH | Edge | 1059$_H$ | | | +164$_H$ |
| 90 | EIC90 | FFFF B0B4$_H$ | INTCSIH1TIRE | Transmission Interrupt Reception Error signal for CSIH_1 | CSIH | Edge | 105A$_H$ | | | +168$_H$ |
| 91 | EIC91 | FFFF B0B6$_H$ | INTCSIH1TIR | Transmission Interrupt Reception signal for CSIH_1 | CSIH | Edge | 105B$_H$ | | | +16C$_H$ |
| 92 | EIC92 | FFFF B0B8$_H$ | INTCSIH1TIC | Transmission Interrupt Communication signal for CSIH_1 | CSIH | Edge | 105C$_H$ | | | +170$_H$ |
| 93 | EIC93 | FFFF B0BA$_H$ | INTCSIH1TIJC | Transmission Interrupt for JOB Completion for CSIH_1 | CSIH | Edge | 105D$_H$ | | | +174$_H$ |
| 94 | EIC94 | FFFF B0BC$_H$ | INTCSIH2TIRE | Transmission Interrupt Reception Error signal for CSIH_2 | CSIH | Edge | 105E$_H$ | | | +178$_H$ |
| 95 | EIC95 | FFFF B0BE$_H$ | INTCSIH2TIR | Transmission Interrupt Reception signal for CSIH_2 | CSIH | Edge | 105F$_H$ | | | +17C$_H$ |
| 96 | EIC96 | FFFF B0C0$_H$ | INTCSIH2TIC | Transmission Interrupt Communication signal for CSIH_2 | CSIH | Edge | 1060$_H$ | | | +180$_H$ |
| 97 | EIC97 | FFFF B0C2$_H$ | INTCSIH2TIJC | Transmission Interrupt for JOB Completion for CSIH_2 | CSIH | Edge | 1061$_H$ | | | +184$_H$ |
| 98 | EIC98 | FFFF B0C4$_H$ | INTCSIH3TIRE | Transmission Interrupt Reception Error signal for CSIH_3 | CSIH | Edge | 1062$_H$ | | | +188$_H$ |
| 99 | EIC99 | FFFF B0C6$_H$ | INTCSIH3TIR | Transmission Interrupt Reception signal for CSIH_3 | CSIH | Edge | 1063$_H$ | | | +18C$_H$ |
| 100 | EIC100 | FFFF B0C8$_H$ | INTCSIH3TIC | Transmission Interrupt Communication signal for CSIH_3 | CSIH | Edge | 1064$_H$ | | | +190$_H$ |
| 101 | EIC101 | FFFF B0CA$_H$ | INTCSIH3TIJC | Transmission Interrupt for JOB Completion for CSIH_3 | CSIH | Edge | 1065$_H$ | | | +194$_H$ |
| 102 | EIC102 | FFFF B0CC$_H$ | INTRLIN30UR2 | RLIN3_0 interrupt 0 status | RLIN3 | Edge | 1066$_H$ | | | +198$_H$ |
| 103 | EIC103 | FFFF B0CE$_H$ | INTRLIN30UR1 | RLIN3_0 interrupt 0 receive | RLIN3 | Edge | 1067$_H$ | | | +19C$_H$ |
| 104 | EIC104 | FFFF B0D0$_H$ | INTRLIN30UR0 | RLIN3_0 interrupt 0 transfer | RLIN3 | Edge | 1068$_H$ | | | +1A0$_H$ |
| 105 | EIC105 | FFFF B0D2$_H$ | INTRLIN31UR2 | RLIN3_1 interrupt 0 status | RLIN3 | Edge | 1069$_H$ | | | +1A4$_H$ |
| 106 | EIC106 | FFFF B0D4$_H$ | INTRLIN31UR1 | RLIN3_1 interrupt 0 receive | RLIN3 | Edge | 106A$_H$ | | | +1A8$_H$ |
| 107 | EIC107 | FFFF B0D6$_H$ | INTRLIN31UR0 | RLIN3_1 interrupt 0 transfer | RLIN3 | Edge | 106B$_H$ | | | +1AC$_H$ |
| 108 | EIC108 | FFFF B0D8$_H$ | INTRLIN32UR2 | RLIN3_2 interrupt 0 status | RLIN3 | Edge | 106C$_H$ | | | +1B0$_H$ |
| 109 | EIC109 | FFFF B0DA$_H$ | INTRLIN32UR1 | RLIN3_2 interrupt 0 receive | RLIN3 | Edge | 106D$_H$ | | | +1B4$_H$ |
| 110 | EIC110 | FFFF B0DC$_H$ | INTRLIN32UR0 | RLIN3_2 interrupt 0 transfer | RLIN3 | Edge | 106E$_H$ | | | +1B8$_H$ |
| 111 | EIC111 | FFFF B0DE$_H$ | INTRLIN33UR2 | RLIN3_3 interrupt 0 status | RLIN3 | Edge | 106F$_H$ | | | +1BC$_H$ |
| 112 | EIC112 | FFFF B0E0$_H$ | INTRLIN33UR1 | RLIN3_3 interrupt 0 receive | RLIN3 | Edge | 1070$_H$ | | | +1C0$_H$ |
| 113 | EIC113 | FFFF B0E2$_H$ | INTRLIN33UR0 | RLIN3_3 interrupt 0 transfer | RLIN3 | Edge | 1071$_H$ | | | +1C4$_H$ |
| 114 | Reserved | | | | | | 1072$_H$ | | | +1C8$_H$ |

**Table 6.5 RH850/P1x-C EI level maskable interrupt sources (4/7)**

| Interrupt | | | Interrupt request | | | Detection type*2 | Exception code | Handler address (offset)*6 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Control register | | | | | | | Direct jumping to an address | | |
| Channel*1 | Name | Address | Name | Cause | Unit | | | RINT = 0*3 | RINT = 1*4 | Reference to a table*5 |
| 115 | Reserved | | | | | | $1073_H$ | Offset address does not differ from channel to channel; it is determined according to the priority order, and is taken from an address between $+100_H$ to $+1F0_H$. | Offset address is uniformly $+100_H$ irrespective of the priority order. | $+1CC_H$ |
| 116 | EIC116 | FFFF B0E8$_H$ | INTHSUS0ERR | Bundled error interrupt for HS-USRT0 | HS-USART | Edge | $1074_H$ | | | $+1D0_H$ |
| 117 | EIC117 | FFFFB0EA$_H$ | INTHSUS0RE | Bundled reception interrupt for HS-USRT0 | HS-USART | Edge | $1075_H$ | | | $+1D4_H$ |
| 118 | EIC118 | FFFF B0EC$_H$ | INTHSUS0TR | Bundled transmission interrupt for HS-USRT0 | HS-USART | Edge | $1076_H$ | | | $+1D8_H$ |
| 119 | EIC119 | FFFF B0EE$_H$ | INTHSUS1ERR | Bundled error interrupt for HS-USRT1 | HS-USART | Edge | $1077_H$ | | | $+1DC_H$ |
| 120 | EIC120 | FFFF B0F0$_H$ | INTHSUS1RE | Bundled reception interrupt for HS-USRT1 | HS-USART | Edge | $1078_H$ | | | $+1E0_H$ |
| 121 | EIC121 | FFFF B0F2$_H$ | INTHSUS1TR | Bundled transmission interrupt for HS-USRT1 | HS-USART | Edge | $1079_H$ | | | $+1E4_H$ |
| 122 | EIC122 | FFFF B0F4$_H$ | INTHSUS2ERR | Bundled error interrupt for HS-USRT2 | HS-USART | Edge | $107A_H$ | | | $+1E8_H$ |
| 123 | EIC123 | FFFF B0F6$_H$ | INTHSUS2RE | Bundled reception interrupt for HS-USRT2 | HS-USART | Edge | $107B_H$ | | | $+1EC_H$ |
| 124 | EIC124 | FFFF B0F8$_H$ | INTHSUS2TR | Bundled transmission interrupt for HS-USRT2 | HS-USART | Edge | $107C_H$ | | | $+1F0_H$ |
| 125 | EIC125 | FFFF B0FA$_H$ | INTHSUS3ERR | Bundled error interrupt for HS-USRT3 | HS-USART | Edge | $107D_H$ | | | $+1F4_H$ |
| 126 | EIC126 | FFFF B0FC$_H$ | INTHSUS3RE | Bundled reception interrupt for HS-USRT3 | HS-USART | Edge | $107E_H$ | | | $+1F8_H$ |
| 127 | EIC127 | FFFF B0FE$_H$ | INTHSUS3TR | Bundled transmission interrupt for HS-USRT3 | HS-USART | Edge | $107F_H$ | | | $+1FC_H$ |
| 128 | EIC128 | FFFF B100$_H$ | INTP5 | External Interrupt5 | Pin | Edge | $1080_H$ | | | $+200_H$ |
| 129 | EIC129 | FFFF B102$_H$ | INTP6 | External Interrupt6 | Pin | Edge | $1081_H$ | | | $+204_H$ |
| 130 | EIC130 | FFFF B104$_H$ | INTP7 | External Interrupt7 | Pin | Edge | $1082_H$ | | | $+208_H$ |
| 131 | EIC131 | FFFF B106$_H$ | INTP8 | External Interrupt8 | Pin | Edge | $1083_H$ | | | $+20C_H$ |
| 132 | EIC132 | FFFF B108$_H$ | INTP9 | External Interrupt9 | Pin | Edge | $1084_H$ | | | $+210_H$ |
| 133 | EIC133 | FFFF B10A$_H$ | INTP10 | External Interrupt10 | Pin | Edge | $1085_H$ | | | $+214_H$ |
| 134 | EIC134 | FFFF B10C$_H$ | INTP11 | External Interrupt11 | Pin | Edge | $1086_H$ | | | $+218_H$ |
| 135 | EIC135 | FFFF B10E$_H$ | INTSTM06 | STM interrupt 6 (STM0) | STM | Edge | $1087_H$ | | | $+21C_H$ |
| 136 | EIC136 | FFFF B110$_H$ | INTSTM07 | STM interrupt 7 (STM0) | STM | Edge | $1088_H$ | | | $+220_H$ |
| 137 | EIC137 | FFFF B112$_H$ | INTSTM14 | STM interrupt 4 (STM1) | STM | Edge | $1089_H$ | | | $+224_H$ |
| 138 | EIC138 | FFFF B114$_H$ | INTSTM15 | STM interrupt 5 (STM1) | STM | Edge | $108A_H$ | | | $+228_H$ |
| 139 | EIC139 | FFFF B116$_H$ | INTSTM16 | STM interrupt 6 (STM1) | STM | Edge | $108B_H$ | | | $+22C_H$ |
| 140 | EIC140 | FFFF B118$_H$ | INTSTM17 | STM interrupt 7 (STM1) | STM | Edge | $108C_H$ | | | $+230_H$ |
| 141 | EIC141 | FFFF B11A$_H$ | INTGTM0TIM01 | TIM Shared interrupts (TIM0_IRQ1) | GTM | Edge | $108D_H$ | | | $+234_H$ |
| 142 | EIC142 | FFFF B11C$_H$ | INTGTM0TIM03 | TIM Shared interrupts (TIM0_IRQ3) | GTM | Edge | $108E_H$ | | | $+238_H$ |
| 143 | EIC143 | FFFF B11E$_H$ | INTGTM0TIM05 | TIM Shared interrupts (TIM0_IRQ5) | GTM | Edge | $108F_H$ | | | $+23C_H$ |
| 144 | EIC144 | FFFF B120$_H$ | INTGTM0TIM07 | TIM Shared interrupts (TIM0_IRQ7) | GTM | Edge | $1090_H$ | | | $+240_H$ |
| 145 | EIC145 | FFFF B122$_H$ | INTGTM0TIM11 | TIM Shared interrupts (TIM1_IRQ1) | GTM | Edge | $1091_H$ | | | $+244_H$ |
| 146 | EIC146 | FFFF B124$_H$ | INTGTM0TIM13 | TIM Shared interrupts (TIM1_IRQ3) | GTM | Edge | $1092_H$ | | | $+248_H$ |
| 147 | EIC147 | FFFF B126$_H$ | INTGTM0TIM15 | TIM Shared interrupts (TIM1_IRQ5) | GTM | Edge | $1093_H$ | | | $+24C_H$ |
| 148 | EIC148 | FFFF B128$_H$ | INTGTM0TIM17 | TIM Shared interrupts (TIM1_IRQ7) | GTM | Edge | $1094_H$ | | | $+250_H$ |
| 149 | EIC149 | FFFF B12A$_H$ | INTGTM0MCS01 | MCS Interrupt for channel (MCS0_IRQ1) | GTM | Edge | $1095_H$ | | | $+254_H$ |

Table 6.5 RH850/P1x-C EI level maskable interrupt sources (5/7)

| Interrupt | | | Interrupt request | | | Detection type*2 | Exception code | Handler address (offset)*6 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Control register | | | | | | | Direct jumping to an address | | |
| Channel*1 | Name | Address | Name | Cause | Unit | | | RINT = 0*3 | RINT = 1*4 | Reference to a table*5 |
| 150 | EIC150 | FFFF B12C$_H$ | INTGTM0MCS03 | MCS Interrupt for channel (MCS0_IRQ3) | GTM | Edge | 1096$_H$ | Offset address does not differ from channel to channel; it is determined according to the priority order, and is taken from an address between +100$_H$ to +1F0$_H$. | Offset address is uniformly +100$_H$ irrespective of the priority order. | +258$_H$ |
| 151 | EIC151 | FFFF B12E$_H$ | INTGTM0MCS05 | MCS Interrupt for channel (MCS0_IRQ5) | GTM | Edge | 1097$_H$ | | | +25C$_H$ |
| 152 | EIC152 | FFFF B130$_H$ | INTGTM0MCS07 | MCS Interrupt for channel (MCS0_IRQ7) | GTM | Edge | 1098$_H$ | | | +260$_H$ |
| 153 | EIC153 | FFFF B132$_H$ | INTGTM0MCS10 | MCS Interrupt for channel (MCS1_IRQ0) | GTM | Edge | 1099$_H$ | | | +264$_H$ |
| 154 | EIC154 | FFFF B134$_H$ | INTGTM0MCS12 | MCS Interrupt for channel (MCS1_IRQ2) | GTM | Edge | 109A$_H$ | | | +268$_H$ |
| 155 | EIC155 | FFFF B136$_H$ | INTGTM0MCS14 | MCS Interrupt for channel (MCS1_IRQ4) | GTM | Edge | 109B$_H$ | | | +26C$_H$ |
| 156 | EIC156 | FFFF B138$_H$ | INTGTM0ATOM01 | ATOM Shared interrupts (ATOM0_IRQ1) | GTM | Edge | 109C$_H$ | | | +270$_H$ |
| 157 | EIC157 | FFFF B13A$_H$ | INTGTM0ATOM03 | ATOM Shared interrupts (ATOM0_IRQ3) | GTM | Edge | 109D$_H$ | | | +274$_H$ |
| 158 | EIC158 | FFFF B13C$_H$ | INTGTM0ATOM05 | ATOM Shared interrupts (ATOM0_IRQ5) | GTM | Edge | 109E$_H$ | | | +278$_H$ |
| 159 | EIC159 | FFFF B13E$_H$ | INTGTM0ATOM07 | ATOM Shared interrupts (ATOM0_IRQ7) | GTM | Edge | 109F$_H$ | | | +27C$_H$ |
| 160 | EIC160 | FFFF B140$_H$ | INTGTM0ATOM11 | ATOM Shared interrupts (ATOM1_IRQ1) | GTM | Edge | 10A0$_H$ | | | +280$_H$ |
| 161 | EIC161 | FFFF B142$_H$ | INTGTM0ATOM13 | ATOM Shared interrupts (ATOM1_IRQ3) | GTM | Edge | 10A1$_H$ | | | +284$_H$ |
| 162 | EIC162 | FFFF B144$_H$ | INTGTM0ATOM15 | ATOM Shared interrupts (ATOM1_IRQ5) | GTM | Edge | 10A2$_H$ | | | +288$_H$ |
| 163 | EIC163 | FFFF B146$_H$ | INTGTM0ATOM17 | ATOM Shared interrupts (ATOM1_IRQ7) | GTM | Edge | 10A3$_H$ | | | +28C$_H$ |
| 164 | EIC164 | FFFF B148$_H$ | INTGTM0ATOM21 | ATOM Shared interrupts (ATOM2_IRQ1) | GTM | Edge | 10A4$_H$ | | | +290$_H$ |
| 165 | EIC165 | FFFF B14A$_H$ | INTGTM0ATOM23 | ATOM Shared interrupts (ATOM2_IRQ3) | GTM | Edge | 10A5$_H$ | | | +294$_H$ |
| 166 | EIC166 | FFFF B14C$_H$ | INTADCF1ERR | Error Interrupt for ADCF1 | ADC | Edge | 10A6$_H$ | | | +298$_H$ |
| 167 | EIC167 | FFFF B14E$_H$ | INTADCF1I0 | Trigger group for SG0 of ADCF1 | ADC | Edge | 10A7$_H$ | | | +29C$_H$ |
| 168 | EIC168 | FFFF B150$_H$ | INTADCF1I1 | Trigger group for SG1 of ADCF1 | ADC | Edge | 10A8$_H$ | | | +2A0$_H$ |
| 169 | EIC169 | FFFF B152$_H$ | INTADCF1I2 | Trigger group for SG2 of ADCF1 | ADC | Edge | 10A9$_H$ | | | +2A4$_H$ |
| 170 | EIC170 | FFFF B154$_H$ | INTADCF1I3 | Trigger group for SG3 of ADCF1 | ADC | Edge | 10AA$_H$ | | | +2A8$_H$ |
| 171 | EIC171 | FFFF B156$_H$ | INTADCF1I4 | Trigger group for SG4 of ADCF1 | ADC | Edge | 10AB$_H$ | | | +2AC$_H$ |
| 172 | EIC172 | FFFF B158$_H$ | INTMTTCANI0 | Interrupt0 for MTTCAN0 | MCAN | Level | 10AC$_H$ | | | +2B0$_H$ |
| 173 | EIC173 | FFFF B15A$_H$ | INTMTTCANI1 | Interrupt1 for MTTCAN0 | MCAN | Level | 10AD$_H$ | | | +2B4$_H$ |
| 174 | EIC174 | FFFF B15C$_H$ | INTMTTCANFE | Filter event for MTTCAN0 | MCAN | Edge | 10AE$_H$ | | | +2B8$_H$ |
| 175 | EIC175 | FFFF B15E$_H$ | INTMCAN0I0 | Interrupt0 for MCAN0 | MCAN | Level | 10AF$_H$ | | | +2BC$_H$ |
| 176 | EIC176 | FFFF B160$_H$ | INTMCAN0I1 | Interrupt1 for MCAN0 | MCAN | Level | 10B0$_H$ | | | +2C0$_H$ |
| 177 | EIC177 | FFFF B162$_H$ | INTMCAN0FE | Filter event for MCAN0 | MCAN | Edge | 10B1$_H$ | | | +2C4$_H$ |
| 178 | EIC178 | FFFF B164$_H$ | INTMCAN1I0 | Interrupt0 for MCAN1 | MCAN | Level | 10B2$_H$ | | | +2C8$_H$ |
| 179 | EIC179 | FFFF B166$_H$ | INTMCAN1I1 | Interrupt1 for MCAN1 | MCAN | Level | 10B3$_H$ | | | +2CC$_H$ |
| 180 | EIC180 | FFFF B168$_H$ | INTMCAN1FE | Filter event for MCAN1 | MCAN | Edge | 10B4$_H$ | | | +2D0$_H$ |
| 181 | EIC181 | FFFF B16A$_H$ | INTMCAN2I0 | Interrupt0 for MCAN2 | MCAN | Level | 10B5$_H$ | | | +2D4$_H$ |
| 182 | EIC182 | FFFF B16C$_H$ | INTMCAN2I1 | Interrupt1 for MCAN2 | MCAN | Level | 10B6$_H$ | | | +2D8$_H$ |
| 183 | EIC183 | FFFF B16E$_H$ | INTMCAN2FE | Filter event for MCAN2 | MCAN | Edge | 10B7$_H$ | | | +2DC$_H$ |
| 184 | EIC184 | FFFF B170$_H$ | INTETNA0 | Ethernet interrupt for ETNA0 | Ethernet | Level | 10B8$_H$ | | | +2E0$_H$ |
| 185 | EIC185 | FFFF B172$_H$ | INTETNA1 | Ethernet interrupt for ETNA1 | Ethernet | Level | 10B9$_H$ | | | +2E4$_H$ |
| 186 | EIC186 | FFFF B174$_H$ | INTFLX0LINE0 | Universal interrupt chA for FLXA0 | FlexRay | Level | 10BA$_H$ | | | +2E8$_H$ |
| 187 | EIC187 | FFFF B176$_H$ | INTFLX0LINE1 | Universal interrupt chB for FLXA0 | FlexRay | Level | 10BB$_H$ | | | +2EC$_H$ |
| 188 | EIC188 | FFFF B178$_H$ | INTFLX0TIM0 | Timer 0 interrupt for FLXA0 | FlexRay | Level | 10BC$_H$ | | | +2F0$_H$ |
| 189 | EIC189 | FFFF B17A$_H$ | INTFLX0TIM1 | Timer 1 interrupt for FLXA0 | FlexRay | Level | 10BD$_H$ | | | +2F4$_H$ |
| 190 | EIC190 | FFFF B17C$_H$ | INTFLX0TIM2 | Timer 2 interrupt for FLXA0 | FlexRay | Level | 10BE$_H$ | | | +2F8$_H$ |
| 191 | EIC191 | FFFF B17E$_H$ | INTFLX0FDA | FIFO data available(FIFO not empty) interrupt for FLXA0 | FlexRay | Level | 10BF$_H$ | | | +2FC$_H$ |
| 192 | EIC192 | FFFF B180$_H$ | INTFLX0FW | FIFO warning interrupt for FLXA0 | FlexRay | Level | 10C0$_H$ | | | +300$_H$ |
| 193 | EIC193 | FFFF B182$_H$ | INTFLX0OW | Output transfer warning interrupt for FLXA0 | FlexRay | Level | 10C1$_H$ | | | +304$_H$ |

**Table 6.5 RH850/P1x-C EI level maskable interrupt sources (6/7)**

| Interrupt | | | Interrupt request | | | Detection type*2 | Exception code | Handler address (offset)*6 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Control register | | | | | | | Direct jumping to an address | | |
| Channel*1 | Name | Address | Name | Cause | Unit | | | RINT = 0*3 | RINT = 1*4 | Reference to a table*5 |
| 194 | EIC194 | FFFF B184$_H$ | INTFLX0OT | Output transfer done interrupt for FLXA0 | FlexRay | Level | 10C2$_H$ | Offset address does not differ from channel to channel; it is determined according to the priority order, and is taken from an address between +100$_H$ to +1F0$_H$. | Offset address is uniformly +100$_H$ irrespective of the priority order. | +308$_H$ |
| 195 | EIC195 | FFFF B186$_H$ | INTFLX0IQF | Input queue full interrupt for FLXA0 | FlexRay | Level | 10C3$_H$ | | | +30C$_H$ |
| 196 | EIC196 | FFFF B188$_H$ | INTFLX0IQE | Input queue empty interrupt for FLXA0 | FlexRay | Level | 10C4$_H$ | | | +310$_H$ |
| 197 | EIC197 | FFFF B18A$_H$ | INTFLX1LINE0 | Universal interrupt chA for FLXA1 | FlexRay | Level | 10C5$_H$ | | | +314$_H$ |
| 198 | EIC198 | FFFF B18C$_H$ | INTFLX1LINE1 | Universal interrupt chB for FLXA1 | FlexRay | Level | 10C6$_H$ | | | +318$_H$ |
| 199 | EIC199 | FFFF B18E$_H$ | INTFLX1TIM0 | Timer 0 interrupt for FLXA1 | FlexRay | Level | 10C7$_H$ | | | +31C$_H$ |
| 200 | EIC200 | FFFF B190$_H$ | INTFLX1TIM1 | Timer 1 interrupt for FLXA1 | FlexRay | Level | 10C8$_H$ | | | +320$_H$ |
| 201 | EIC201 | FFFF B192$_H$ | INTFLX1TIM2 | Timer 2 interrupt for FLXA1 | FlexRay | Level | 10C9$_H$ | | | +324$_H$ |
| 202 | EIC202 | FFFF B194$_H$ | INTFLX1FDA | FIFO data available (FIFO not empty) interrupt for FLXA1 | FlexRay | Level | 10CA$_H$ | | | +328$_H$ |
| 203 | EIC203 | FFFF B196$_H$ | INTFLX1FW | FIFO warning interrupt for FLXA1 | FlexRay | Level | 10CB$_H$ | | | +32C$_H$ |
| 204 | EIC204 | FFFF B198$_H$ | INTFLX1OW | Output transfer warning interrupt for FLXA1 | FlexRay | Level | 10CC$_H$ | | | +330$_H$ |
| 205 | EIC205 | FFFF B19A$_H$ | INTFLX1OT | Output transfer done interrupt for FLXA1 | FlexRay | Level | 10CD$_H$ | | | +334$_H$ |
| 206 | EIC206 | FFFF B19C$_H$ | INTFLX1IQF | Input queue full interrupt for FLXA1 | FlexRay | Level | 10CE$_H$ | | | +338$_H$ |
| 207 | EIC207 | FFFF B19E$_H$ | INTFLX1IQE | Input queue empty interrupt for FLXA1 | FlexRay | Level | 10CF$_H$ | | | +33C$_H$ |
| 208 | EIC208 | FFFF B1A0$_H$ | INTSENT0SI | status interrupt for RSENT0 | RSENT | Level | 10D0$_H$ | | | +340$_H$ |
| 209 | EIC209 | FFFF B1A2$_H$ | INTSENT0RI | receive interrupt for RSENT0 | RSENT | Edge | 10D1$_H$ | | | +344$_H$ |
| 210 | EIC210 | FFFF B1A4$_H$ | INTSENT1SI | status interrupt for RSENT1 | RSENT | Level | 10D2$_H$ | | | +348$_H$ |
| 211 | EIC211 | FFFF B1A6$_H$ | INTSENT1RI | receive interrupt for RSENT1 | RSENT | Edge | 10D3$_H$ | | | +34C$_H$ |
| 212 | EIC212 | FFFF B1A8$_H$ | INTSENT2SI | status interrupt for RSENT2 | RSENT | Level | 10D4$_H$ | | | +350$_H$ |
| 213 | EIC213 | FFFF B1AA$_H$ | INTSENT2RI | receive interrupt for RSENT2 | RSENT | Edge | 10D5$_H$ | | | +354$_H$ |
| 214 | EIC214 | FFFF B1AC$_H$ | INTSENT3SI | status interrupt for RSENT3 | RSENT | Level | 10D6$_H$ | | | +358$_H$ |
| 215 | EIC215 | FFFF B1AE$_H$ | INTSENT3RI | receive interrupt for RSENT3 | RSENT | Edge | 10D7$_H$ | | | +35C$_H$ |
| 216 | EIC216 | FFFF B1B0$_H$ | INTSENT4SI | status interrupt for RSENT4 | RSENT | Level | 10G8$_H$ | | | +360$_H$ |
| 217 | EIC217 | FFFF B1B2$_H$ | INTSENT4RI | receive interrupt for RSENT4 | RSENT | Edge | 10D9$_H$ | | | +364$_H$ |
| 218 | EIC218 | FFFF B1B4$_H$ | INTSENT5SI | status interrupt for RSENT5 | RSENT | Level | 10DA$_H$ | | | +368$_H$ |
| 219 | EIC219 | FFFF B1B6$_H$ | INTSENT5RI | receive interrupt for RSENT5 | RSENT | Edge | 10DB$_H$ | | | +36C$_H$ |
| 220 | EIC220 | FFFF B1B8$_H$ | INTSENT6SI | status interrupt for RSENT6 | RSENT | Level | 10DC$_H$ | | | +370$_H$ |
| 221 | EIC221 | FFFF B1BA$_H$ | INTSENT6RI | receive interrupt for RSENT6 | RSENT | Edge | 10DD$_H$ | | | +374$_H$ |
| 222 | EIC222 | FFFF B1BC$_H$ | INTSENT7SI | status interrupt for RSENT7 | RSENT | Level | 10DE$_H$ | | | +378$_H$ |
| 223 | EIC223 | FFFF B1BE$_H$ | INTSENT7RI | receive interrupt for RSENT7 | RSENT | Edge | 10DF$_H$ | | | +37C$_H$ |
| 224 | EIC224 | FFFF B1C0$_H$ | INTSENT8SI | status interrupt for RSENT8 | RSENT | Level | 10E0$_H$ | | | +380$_H$ |
| 225 | EIC225 | FFFF B1C2$_H$ | INTSENT8RI | receive interrupt for RSENT8 | RSENT | Edge | 10E1$_H$ | | | +384$_H$ |
| 226 | EIC226 | FFFF B1C4$_H$ | INTSENT9SI | status interrupt for RSENT9 | RSENT | Level | 10E2$_H$ | | | +388$_H$ |
| 227 | EIC227 | FFFF B1C6$_H$ | INTSENT9RI | receive interrupt for RSENT9 | RSENT | Edge | 10E3$_H$ | | | +38C$_H$ |
| 228 | EIC228 | FFFF B1C8$_H$ | INTDTSTC0 | DTS transmission complete interrupt for ch0 to 31 | DTS | Level | 10E4$_H$ | | | +390$_H$ |
| 229 | EIC229 | FFFF B1CA$_H$ | INTDTSTC1 | DTS transmission complete interrupt for ch32 to 63 | DTS | Level | 10E5$_H$ | | | +394$_H$ |
| 230 | EIC230 | FFFF B1CC$_H$ | INTDTSTC2 | DTS transmission complete interrupt for ch64 to 95 | DTS | Level | 10E6$_H$ | | | +398$_H$ |
| 231 | EIC231 | FFFF B1CE$_H$ | INTDTSTC3 | DTS transmission complete interrupt for ch96 to 127 | DTS | Level | 10E7$_H$ | | | +39C$_H$ |
| 232 | EIC232 | FFFF B1D0$_H$ | INTDTSCM0 | DTS count match interrupt for ch0 to 31 | DTS | Level | 10E8$_H$ | | | +3A0$_H$ |
| 233 | EIC233 | FFFFB1D2$_H$ | INTDTSCM1 | DTS count match interrupt for c32 to 63 | DTS | Level | 10E9$_H$ | | | +3A4$_H$ |
| 234 | EIC234 | FFFF B1D4$_H$ | INTDTSCM2 | DTS count match interrupt for ch64 to 95 | DTS | Level | 10EA$_H$ | | | +3A8$_H$ |
| 235 | EIC235 | FFFF B1D6$_H$ | INTDTSCM3 | DTS count match interrupt for ch96 to 127 | DTS | Level | 10EB$_H$ | | | +3AC$_H$ |

**Table 6.5 RH850/P1x-C EI level maskable interrupt sources (7/7)**

| Interrupt | | | Interrupt request | | | Detection type*2 | Exception code | Handler address (offset)*6 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Control register | | | | | | | Direct jumping to an address | | |
| Channel*1 | Name | Address | Name | Cause | Unit | | | RINT = 0*3 | RINT = 1*4 | Reference to a table*5 |
| 236 | EIC236 | FFFF B1D8$_H$ | INTDCUDEGRADPE | interrupt for degradation | BHP | Level | 10EC$_H$ | Offset address does not differ from channel to channel; it is determined according to the priority order, and is taken from an address between +100$_H$ to +1F0$_H$. | Offset address is uniformly +100$_H$ irrespective of the priority order. | +3B0$_H$ |
| 237 | EIC237 | FFFF B1DA$_H$ | INTFICUIF0CFRQPE | FENTRYC bit update interrupt for FICUIF_0 | FICUIF | Edge | 10ED$_H$ | | | +3B4$_H$ |
| 238 | EIC238 | FFFF B1DC$_H$ | INTFICUIF0DFRQPE | FENTRYD bit update interrupt for FICUIF_0 | FICUIF | Edge | 10EE$_H$ | | | +3B8$_H$ |
| 239 | EIC239 | FFFF B1DE$_H$ | INTFICUIF1CFRQPE | FENTRYC bit update interrupt for FICUIF_1 | FICUIF | Edge | 10EF$_H$ | | | +3BC$_H$ |
| 240 | EIC240 | FFFF B1E0$_H$ | INTDNFA2WUF0 | Edge detect of RLIN3 0 received data | Pin | Edge | 10F0$_H$ | | | +3C0$_H$ |
| 241 | EIC241 | FFFF B1E2$_H$ | INTDNFA2WUF1 | Edge detect of RLIN3 1 received data | Pin | Edge | 10F1$_H$ | | | +3C4$_H$ |
| 242 | EIC242 | FFFF B1E4$_H$ | INTDNFA3WUF0 | Edge detect of MTTCAN0 received data | Pin | Edge | 10F2$_H$ | | | +3C8$_H$ |
| 243 | EIC243 | FFFF B1E6$_H$ | INTDNFA3WUF1 | Edge detect of MCAN0 received data | Pin | Edge | 10F3$_H$ | | | +3CC$_H$ |
| 244 | EIC244 | FFFF B1E8$_H$ | INTDNFA4WUF0 | Edge detect of FlexRay 0 ch A received data | Pin | Edge | 10F4$_H$ | | | +3D0$_H$ |
| 245 | EIC245 | FFFF B1EA$_H$ | INTDNFA4WUF1 | Edge detect of FlexRay 1 ch A received data | Pin | Edge | 10F5$_H$ | | | +3D4$_H$ |
| 246 | Reserved | | | | | | 10F6$_H$ | | | +3D8$_H$ |
| 247 | Reserved | | | | | | 10F7$_H$ | | | +3DC$_H$ |
| 248 | Reserved | | | | | | 10F8$_H$ | | | +3E0$_H$ |
| 249 | Reserved | | | | | | 10F9$_H$ | | | +3E4$_H$ |
| 250 | Reserved | | | | | | 10FA$_H$ | | | +3E8$_H$ |
| 251 | Reserved | | | | | | 10FB$_H$ | | | +3EC$_H$ |
| 252 | Reserved | | | | | | 10FC$_H$ | | | +3F0$_H$ |
| 253 | Reserved | | | | | | 10FD$_H$ | | | +3F4$_H$ |
| 254 | Reserved | | | | | | 10FE$_H$ | | | +3F8$_H$ |
| 255 | Reserved | | | | | | 10FF$_H$ | | | +3FC$_H$ |

Note 1. 0 to 31: Connected to INTC1. 32 to 255: Connected to INTC2

Note 2. This indicates whether an interrupt source is detected at the level or edge. This also affects the initial value of an EI level interrupt control register. For details, see **Section 6.4.1, EICn — EI level interrupt control registers**.

Note 3. Irrespective of interrupt channels, an offset address is determined according to the priority order (0 to 7), from the range between +100$_H$ to 1F0$_H$.

Note 4. Irrespective of the priority order, offset addresses are uniformly +100$_H$.

Note 5. The table reference method has a table that reads exception handler address for each interrupt channel, and it extracts handler address by referencing that table. Table reference position is determined by the following formula.
Exception handler address read position = INTBP register + channel number × 4 bytes

Note 6. For details, see **Section 6.9, Exception Handler Address**.

## 6.3 Edge Detection Configuration

The external interrupts INTPm and NMI can be configured to generate an interrupt request upon a rising or falling edge or upon both edges of the external pin.

The following registers are used to specify the edge and level of each interrupt:

**Table 6.6 External interrupt edge detection registers**

| Interrupt | Register |
|---|---|
| INTP0 | FCLA0CTL1 |
| INTP1 | FCLA0CTL2 |
| INTP2 | FCLA0CTL3 |
| INTP3 | FCLA0CTL4 |
| INTP4 | FCLA0CTL5 |
| INTP5 | FCLA0CTL6 |
| INTP6 | FCLA0CTL7 |
| INTP7 | FCLA1CTL0 |
| INTP8 | FCLA1CTL1 |
| INTP9 | FCLA1CTL2 |
| INTP10 | FCLA1CTL3 |
| INTP11 | FCLA1CTL4 |
| NMI | FCLA0CTL0 |

See **Section 2, Pin Functions** for details of these registers.

## 6.4 Interrupt Controller Control Registers

Writing to the EICn and IMRm (m = 0 to 7) registers is enabled only in supervisor mode (PSW.UM = 0).

### 6.4.1 EICn — EI level interrupt control registers

These registers, each of which is for a channel of EI level maskable interrupt INT, are used to set the conditions to control each channel.

**Access:** EICn can be read/written in 16-bit units.
EICnH and EICnL can be read/written in 8- or 1-bit units.

**Address:** See **Table 6.5, RH850/P1x-C EI level maskable interrupt sources.**

**Value after reset:** $008F_H$ (edge detection), $808F_H$ (high level detection)
Note 1. This register is initialized by any reset.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CTn | — | — | RFn | — | — | — | — | MKn | TBn | — | — | P3n | P2n | P1n | P0n |
| Value after reset | 0/1*1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R/W | R | R | R | R | R/W | R/W | R | R | R/W | R/W | R/W | R/W |

Note 1. The value after reset differs depending on the detection type of an interrupt (edge detection: 0, level detection: 1). For details, see **Table 6.5, RH850/P1x-C EI level maskable interrupt sources**.

**Table 6.7 EICn register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | CTn | This bit indicates an interrupt channel type. This bit is read-only.<br>0: Detection in synchronization with an edge is currently selected.<br>1: Detection of the high level is currently selected. |
| 14, 13 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 12 | RFn | This is an interrupt request flag.<br>The RFn bit can be written from a program. Setting the RFn bit to 1 generates an EI level maskable interrupt n (INTn), just as when an interrupt request is acknowledged.<br>0: No interrupt request is made (value after reset).<br>1: Interrupt request is made.<br>**Input Interface** / **Operation**<br>Detection in synchronization with an edge (CTn = 0) / This bit is automatically cleared when an interrupt request is acknowledged by the CPU core. It can be set and cleared by software.<br>Detection of the high level (CTn = 1) / This bit cannot be set or cleared by software. It is readable only It is not cleared when an interrupt request is acknowledged by the CPU core. |
| 11 to 8 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 7 | MKn | This is the interrupt request mask bit.<br>When the MKn bit is set, interrupt requests from the channel are masked and, therefore, not issued to the CPU core. The interrupt pending status is not reflected in the ICSR.PMEI bit for any channels that are masked.<br>When the interrupt request from the channel is masked with MKn = 1, the RFn still reflects the interrupt request for the channel and can be polled in software.<br>When the MKn bit is cleared, interrupt requests from the channel are issued to the CPU core for subsequent processing. The state of the MKn bit is also reflected in the corresponding IMRm register.<br>0: Enables interrupt processing<br>1: Disables interrupt processing (value after reset) |

Table 6.7 EICn register contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 | TBn | This bit is used to select the way to determine the interrupt vector.<br>0: Direct jumping to an address determined from the level of priority<br>1: Reference to a table<br>For details on the way to determine the interrupt vector, see *the RH850 Family Users' Manual: Software*. |
| 5, 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | P3n to P0n | These bits specify the interrupt priority as one of 16 levels, with 0 as the highest and 15 as the lowest.<br>When multiple EI level-interrupt requests are made simultaneously, the interrupt from the source with the highest priority setting in these bits is selected and conveyed to the CPU core for servicing first. When the P3n to P0n bits specify the same priority level for simultaneously occurring interrupt requests, the source with the lower channel number takes priority. This order is fixed. |

**CAUTION**

When a channel n is defined as broadcast interrupt (EIBDn.CST = 1), MKn and RFn bits of the EICn register of the channel must be set to 0 after an initial configuration of the channel. In the period when the EIINTn interrupt is enabled, it is prohibited to mask (MKn = 1) an interrupt processing of the channel. When it is necessary to mask a broadcast interrupt, EIC4 to EIC7 registers in INTC1 of each PE can be used to mask the corresponding broadcast interrupt.

**NOTE**

In this product, all EICn registers can be accessed. Accessing these registers does not affect operation.

## 6.4.2 IMRm — EI level interrupt mask registers (m = 0 to 7)

These registers are a collection of the MKn bits of the EICn registers. Each bit of IMRm reflects the setting of the corresponding MKn bit. Setting IMRm is reflected in the corresponding MKn bit.

**Access:** IMRm can be read/written in 32-bit units.
IMRmH and IMRmL can be read/written in 16-bit units.
MRmHH, IMRmHL, IMRmLH, and IMRmLL can be read/written in 8- or 1-bit units.

**Address:** IMR0: FFFE EAF0$_H$, IMR1: FFFF B404$_H$, IMR2: FFFF B408$_H$, IMR3: FFFF B40C$_H$
IMR4: FFFF B410$_H$, IMR5: FFFF B414$_H$, IMR6: FFFF B418$_H$, IMR7: FFFF B41C$_H$

**Value after reset:** FFFF FFFF$_H$ This register is initialized by any reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IMRmEIMKm ×32+31 | IMRmEIMKm ×32+30 | IMRmEIMKm ×32+29 | IMRmEIMKm ×32+28 | IMRmEIMKm ×32+27 | IMRmEIMKm ×32+26 | IMRmEIMKm ×32+25 | IMRmEIMKm ×32+24 | IMRmEIMKm ×32+23 | IMRmEIMKm ×32+22 | IMRmEIMKm ×32+21 | IMRmEIMKm ×32+20 | IMRmEIMKm ×32+19 | IMRmEIMKm ×32+18 | IMRmEIMKm ×32+17 | IMRmEIMKm ×32+16 |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IMRmEIMKm ×32+15 | IMRmEIMKm ×32+14 | IMRmEIMKm ×32+13 | IMRmEIMKm ×32+12 | IMRmEIMKm ×32+11 | IMRmEIMKm ×32+10 | IMRmEIMKm ×32+9 | IMRmEIMKm ×32+8 | IMRmEIMKm ×32+7 | IMRmEIMKm ×32+6 | IMRmEIMKm ×32+5 | IMRmEIMKm ×32+4 | IMRmEIMKm ×32+3 | IMRmEIMKm ×32+2 | IMRmEIMKm ×32+1 | IMRmEIMKm ×32+0 |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 6.8 IMRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | IMRmEIMKm×32+31 to IMRmEIMKm×32 | These are mask bits for EI level maskable interrupt (INT) 0 to 255.<br>0: Enables interrupt servicing<br>1: Disables interrupt servicing |

**CAUTIONS**

1. MKn bits for interrupt channels, not listed in **Table 6.5, RH850/P1x-C EI level maskable interrupt sources** must be set to 1.
2. When a channel is used as broadcast interrupt (EIBDn.CST = 1), IMRmEIMKn bit corresponding to the channel must be set to 0.

### 6.4.3 FNC — FE level NMI status register

This register indicates the status of an FE level non-maskable interrupt (FENMI).

**Access:** FNC can be read in 16-bit units.
FNCH and FNCL can be read in 8- or 1-bit units.

**Address:** FFFE EA78 $_H$

**Value after reset:** $0000_H$ (synchronous edge detection), This register is initialized by any reset.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FNCT | — | — | FNRF | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 6.9 FNC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | FNCT | This bit indicates an interrupt channel type.<br>0: Detection in synchronization with an edge is currently selected.<br>1: Detection of the high level is currently selected. |
| 14, 13 | Reserved | When read, the value after reset is read. |
| 12 | FNRF | Interrupt request flag<br>0: No interrupt request (value after reset)<br>1: Interrupt request occurred |
| 11 to 0 | Reserved | When read, the value after reset is read. |

| Input Interface | Operation |
|---|---|
| Detection in synchronization with an edge (FNCT = 0) | It is automatically cleared when an interrupt request for the FE level NMI is acknowledged by CPU core. This bit can be set and cleared by software in debug mode. |
| Detection of the high level (FNCT = 1) | This bit cannot be set or cleared by software. This bit is read-only. |

### 6.4.4 FIC — FE level maskable interrupt status register

This register indicates the status of an FE level maskable interrupt (FEINT).

**Access:** FIC can be read in 16-bit units.
FICH and FICL can be read in 8- or 1-bit units.

**Address:** FFFE EA7A$_H$

**Value after reset:** 0000$_H$ (synchronous edge detection), This register is initialized by any reset.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FICT | — | — | FIRF | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 6.10 FIC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | FICT | This bit indicates an interrupt channel type. This bit is read-only.<br>0: Detection in synchronization with an edge is currently selected.<br>1: Detection of the high level is currently selected. |
| 14, 13 | Reserved | When read, the value after reset is read. |
| 12 | FIRF | Interrupt request flag<br>0: No interrupt request (value after reset)<br>1: Interrupt request occurred |
| 11 to 0 | Reserved | When read, the value after reset is read. |

| Input Interface | Operation |
|---|---|
| Detection in synchronization with an edge (FICT = 0) | It is automatically cleared when an interrupt request for the FE level NMI is acknowledged by CPU core. This bit can be set and cleared by software in debug mode. |
| Detection of the high level (FICT = 1) | This bit cannot be set or cleared by software. This bit is read-only. |

### 6.4.5 FEINTFn — FE level maskable interrupt event status register

**Access:** FEINTF can be read/written in 32-bit units.

**Address:** FEINTF0: FFD4 $2000_H$, FEINTF1: FFD4 $3000_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by any reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | FEINTESTAT2 | FEINTESTAT1 | FEINTESTAT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 6.11 FEINTF register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is read. |
| 2 | FEINTESTAT2 | Terminal NMI interrupt occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 1 | FEINTESTAT1 | ECM interrupt (INTECMTnNMI) occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |
| 0 | FEINTESTAT0 | STM interrupt 8 occurrence<br>0: No interrupt occurred<br>1: Interrupt has occurred |

### 6.4.6 FEINTFCn — FEINT event clear register

This register clears the bits of the FEINT event register (FEINT).

**Access:** This register can be written in 32-bit units.

**Address:** FEINTFC0: FFD4 $2008_H$, FEINTFC1: FFD4 $3008_H$

**Value after reset:** 0000 $0000_H$
This register is initialized by any reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | FEINTECLR2 | FEINTECLR1 | FEINTECLR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | W | W | W |

**Table 6.12 FEINTFC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When writing, write the value after reset. |
| 2 | FEINTECLR2 | FEINTESTAT2 flag clear<br>0: —<br>1: Cleared |
| 1 | FEINTECLR1 | FEINTESTAT1 flag clear<br>0: —<br>1: Cleared |
| 0 | FEINTECLR0 | FEINTESTAT0 flag clear<br>0: —<br>1: Cleared |

## 6.4.7 FEINTFMSKn — FE level maskable interrupt event mask register

**Access:** FEINTFMSKn can be read/written in 32-bit units.

**Address:** FEINTFMSK0: FFD4 $2004_H$, FEINTFMSK1: FFD4 $3004_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by any reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | FEINTEMSK2 | FEINTEMSK1 | FEINTEMSK0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 6.13 FEINTFMSK register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is read. |
| 2 | FEINTEMSK2 | Terminal NMI interrupt mask<br>1: Interrupt is masked<br>0: Interrupt is not masked |
| 1 | FEINTEMSK1 | ECM interrupt (INTECMTnNMI) mask<br>1: Interrupt is masked<br>0: Interrupt is not masked |
| 0 | FEINTEMSK0 | STM interrupt 8 mask<br>1: Interrupt is masked<br>0: Interrupt is not masked |

## 6.4.8 EIBD0 to EIBD255 — EI Level Interrupt Bind Registers 0 to 255

These registers are provided for each EI level interrupt source to make correspondence between each source and PE.

**Access:** This register can be Read/written in 32-bit units.

**Address:** See **Table 6.15, List of EIBDn register Address**.

**Value after reset:** $0000\ 0001_H$: EIBD0 to 31 (PE1/INTC1)
$0000\ 0002_H$: EIBD0 to 31 (PE2/INTC1)
$0000\ 0001_H$: EIBD32 to 255 (INTC2)
This register is initialized by any reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EIBDnH | CST | — | — | — | — | BCP2n | BCP1n | BCP0n | — | — | — | — | — | GPID 2n | GPID 1n | GPID 0n |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EIBDnL | — | — | — | — | — | — | — | — | — | — | — | — | — | PEID 2n | PEID 1n | PEID 0n |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 6.14 EIBD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | CSTn | By setting CST bit to '1' the corresponding interrupt channel is handled as broadcast interrupt<br>The CST bit can only be written in INTC2 (Interrupt channels EIINT[255:32]).<br>0: Interrupt is handled normally (value after reset)<br>1: Interrupt is defined as broadcast interrupt |
| 30 to 27 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 26 to 24 | BCP[2:0]n | Broadcast Channel selection register<br>These bits are only be used if CST is set to "1"<br>The BCP bits can only be changed in INTC2 (Interrupt channels EIINT[255:32].<br>If CST is set to 0: These bits are ignored<br>If CST is set to 1: The bits BCP[2:0] define the number of the selected broadcast channel |
| 18 to 16 | GPID[2:0]n | These bits are provided only in EIBD32 to EIBD255. Set either of the following values for these bits according to the PEID setting.<br>000: When PE1 is selected by PEID as a bind destination<br>001: When PE2 is selected by PEID as a bind destination<br>These bits are reserved for EIBD0 to EIBD31. The write value should always be 0. These bits are always read as 0. |
| 15 to 3 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 2 to 0 | PEID[2:0]n | These bits specify an interrupt bind (request) destination. However, these bits are fixed in EIBD0 to EIBD31 and cannot be modified.<br>For the INTC2 Interrupt channels EIINT[255:32] the PE can be selected by writing the corresponding PEID to this EIBD register. |

Table 6.15 List of EIBDn register Address (1/2)

| Register name | Address | Register name | Address | Register name | Address | Register name | Address |
|---|---|---|---|---|---|---|---|
| EIBD0 | FFFE EB00$_H$ | EIBD40 | FFFF B8A0 | EIBD80 | FFFF B940$_H$ | EIBD120 | FFFF B9E0$_H$ |
| EIBD1 | FFFE EB04$_H$ | EIBD41 | FFFF B8A4 | EIBD81 | FFFF B944$_H$ | EIBD121 | FFFF B9E4$_H$ |
| EIBD2 | FFFE EB08$_H$ | EIBD42 | FFFF B8A8 | EIBD82 | FFFF B948$_H$ | EIBD122 | FFFF B9E8$_H$ |
| EIBD3 | FFFE EB0C$_H$ | EIBD43 | FFFF B8AC | EIBD83 | FFFF B94C$_H$ | EIBD123 | FFFF B9EC$_H$ |
| EIBD4 | FFFE EB10$_H$ | EIBD44 | FFFF B8B0 | EIBD84 | FFFF B950$_H$ | EIBD124 | FFFF B9F0$_H$ |
| EIBD5 | FFFE EB14$_H$ | EIBD45 | FFFF B8B4 | EIBD85 | FFFF B954$_H$ | EIBD125 | FFFF B9F4$_H$ |
| EIBD6 | FFFE EB18$_H$ | EIBD46 | FFFF B8B8 | EIBD86 | FFFF B958$_H$ | EIBD126 | FFFF B9F8$_H$ |
| EIBD7 | FFFE EB1C$_H$ | EIBD47 | FFFF B8BC | EIBD87 | FFFF B95C$_H$ | EIBD127 | FFFF B9FC$_H$ |
| EIBD8 | FFFE EB20$_H$ | EIBD48 | FFFF B8C0 | EIBD88 | FFFF B960$_H$ | EIBD128 | FFFF BA00$_H$ |
| EIBD9 | FFFE EB24$_H$ | EIBD49 | FFFF B8C4 | EIBD89 | FFFF B964$_H$ | EIBD129 | FFFF BA04$_H$ |
| EIBD10 | FFFE EB28$_H$ | EIBD50 | FFFF B8C8 | EIBD90 | FFFF B968$_H$ | EIBD130 | FFFF BA08$_H$ |
| EIBD11 | FFFE EB2C$_H$ | EIBD51 | FFFF B8CC | EIBD91 | FFFF B96C$_H$ | EIBD131 | FFFF BA0C$_H$ |
| EIBD12 | FFFE EB30$_H$ | EIBD52 | FFFF B8D0 | EIBD92 | FFFF B970$_H$ | EIBD132 | FFFF BA10$_H$ |
| EIBD13 | FFFE EB34$_H$ | EIBD53 | FFFF B8D4$_H$ | EIBD93 | FFFF B974$_H$ | EIBD133 | FFFF BA14$_H$ |
| EIBD14 | FFFE EB38$_H$ | EIBD54 | FFFF B8D8$_H$ | EIBD94 | FFFF B978$_H$ | EIBD134 | FFFF BA18$_H$ |
| EIBD15 | FFFE EB3C$_H$ | EIBD55 | FFFF B8DC$_H$ | EIBD95 | FFFF B97C$_H$ | EIBD135 | FFFF BA1C$_H$ |
| EIBD16 | FFFE EB40$_H$ | EIBD56 | FFFF B8E0$_H$ | EIBD96 | FFFF B980$_H$ | EIBD136 | FFFF BA20$_H$ |
| EIBD17 | FFFE EB44$_H$ | EIBD57 | FFFF B8E4$_H$ | EIBD97 | FFFF B984$_H$ | EIBD137 | FFFF BA24$_H$ |
| EIBD18 | FFFE EB48$_H$ | EIBD58 | FFFF B8E8$_H$ | EIBD98 | FFFF B988$_H$ | EIBD138 | FFFF BA28$_H$ |
| EIBD19 | FFFE EB4C$_H$ | EIBD59 | FFFF B8EC$_H$ | EIBD99 | FFFF B98C$_H$ | EIBD139 | FFFF BA2C$_H$ |
| EIBD20 | FFFE EB50$_H$ | EIBD60 | FFFF B8F0$_H$ | EIBD100 | FFFF B990$_H$ | EIBD140 | FFFF BA30$_H$ |
| EIBD21 | FFFE EB54$_H$ | EIBD61 | FFFF B8F4$_H$ | EIBD101 | FFFF B994$_H$ | EIBD141 | FFFF BA34$_H$ |
| EIBD22 | FFFE EB58$_H$ | EIBD62 | FFFF B8F8$_H$ | EIBD102 | FFFF B998$_H$ | EIBD142 | FFFF BA38$_H$ |
| EIBD23 | FFFE EB5C$_H$ | EIBD63 | FFFF B8FC$_H$ | EIBD103 | FFFF B99C$_H$ | EIBD143 | FFFF BA3C$_H$ |
| EIBD24 | FFFE EB60$_H$ | EIBD64 | FFFF B900$_H$ | EIBD104 | FFFF B9A0$_H$ | EIBD144 | FFFF BA40$_H$ |
| EIBD25 | FFFE EB64$_H$ | EIBD65 | FFFF B904$_H$ | EIBD105 | FFFF B9A4$_H$ | EIBD145 | FFFF BA44$_H$ |
| EIBD26 | FFFE EB68$_H$ | EIBD66 | FFFF B908$_H$ | EIBD106 | FFFF B9A8$_H$ | EIBD146 | FFFF BA48$_H$ |
| EIBD27 | FFFE EB6C$_H$ | EIBD67 | FFFF B90C$_H$ | EIBD107 | FFFF B9AC$_H$ | EIBD147 | FFFF BA4C$_H$ |
| EIBD28 | FFFE EB70$_H$ | EIBD68 | FFFF B910$_H$ | EIBD108 | FFFF B9B0$_H$ | EIBD148 | FFFF BA50$_H$ |
| EIBD29 | FFFE EB74$_H$ | EIBD69 | FFFF B914$_H$ | EIBD109 | FFFF B9B4$_H$ | EIBD149 | FFFF BA54$_H$ |
| EIBD30 | FFFE EB78$_H$ | EIBD70 | FFFF B918$_H$ | EIBD110 | FFFF B9B8$_H$ | EIBD150 | FFFF BA58$_H$ |
| EIBD31 | FFFE EB7C$_H$ | EIBD71 | FFFF B91C$_H$ | EIBD111 | FFFF B9BC$_H$ | EIBD151 | FFFF BA5C$_H$ |
| EIBD32 | FFFF B880$_H$ | EIBD72 | FFFF B920$_H$ | EIBD112 | FFFF B9C0$_H$ | EIBD152 | FFFF BA60$_H$ |
| EIBD33 | FFFF B884$_H$ | EIBD73 | FFFF B924$_H$ | EIBD113 | FFFF B9C4$_H$ | EIBD153 | FFFF BA64$_H$ |
| EIBD34 | FFFF B888$_H$ | EIBD74 | FFFF B928$_H$ | Reserved | FFFF B9C8$_H$ | EIBD154 | FFFF BA68$_H$ |
| EIBD35 | FFFF B88C$_H$ | EIBD75 | FFFF B92C$_H$ | Reserved | FFFF B9CC$_H$ | EIBD155 | FFFF BA6C$_H$ |
| EIBD36 | FFFF B890$_H$ | EIBD76 | FFFF B930$_H$ | EIBD116 | FFFF B9D0$_H$ | EIBD156 | FFFF BA70$_H$ |
| EIBD37 | FFFF B894$_H$ | EIBD77 | FFFF B934$_H$ | EIBD117 | FFFF B9D4$_H$ | EIBD157 | FFFF BA74$_H$ |
| EIBD38 | FFFF B898$_H$ | EIBD78 | FFFF B938$_H$ | EIBD118 | FFFF B9D8$_H$ | EIBD158 | FFFF BA78$_H$ |
| EIBD39 | FFFF B89C$_H$ | EIBD79 | FFFF B93C$_H$ | EIBD119 | FFFF B9DC$_H$ | EIBD159 | FFFF BA7C$_H$ |
| EIBD160 | FFFF BA80$_H$ | EIBD184 | FFFF BAE0$_H$ | EIBD208 | FFFF BB40$_H$ | EIBD232 | FFFF BBA0$_H$ |
| EIBD161 | FFFF BA84$_H$ | EIBD185 | FFFF BAE4$_H$ | EIBD209 | FFFF BB44$_H$ | EIBD233 | FFFF BBA4$_H$ |
| EIBD162 | FFFF BA88$_H$ | EIBD186 | FFFF BAE8$_H$ | EIBD210 | FFFF BB48$_H$ | EIBD234 | FFFF BBA8$_H$ |

Table 6.15 List of EIBDn register Address (2/2)

| Register name | Address | Register name | Address | Register name | Address | Register name | Address |
|---|---|---|---|---|---|---|---|
| EIBD163 | FFFF BA8C$_H$ | EIBD187 | FFFF BAEC$_H$ | EIBD211 | FFFF BB4C$_H$ | EIBD235 | FFFF BBAC$_H$ |
| EIBD164 | FFFF BA90$_H$ | EIBD188 | FFFF BAF0$_H$ | EIBD212 | FFFF BB50$_H$ | EIBD236 | FFFF BBB0$_H$ |
| EIBD165 | FFFF BA94$_H$ | EIBD189 | FFFF BAF4$_H$ | EIBD213 | FFFF BB54$_H$ | EIBD237 | FFFF BBB4$_H$ |
| EIBD166 | FFFF BA98$_H$ | EIBD190 | FFFF BAF8$_H$ | EIBD214 | FFFF BB58$_H$ | EIBD238 | FFFF BBB8$_H$ |
| EIBD167 | FFFF BA9C$_H$ | EIBD191 | FFFF BAFC$_H$ | EIBD215 | FFFF BB5C$_H$ | EIBD239 | FFFF BBBC$_H$ |
| EIBD168 | FFFF BAA0$_H$ | EIBD192 | FFFF BB00$_H$ | EIBD216 | FFFF BB60$_H$ | EIBD240 | FFFF BBC0$_H$ |
| EIBD169 | FFFF BAA4$_H$ | EIBD193 | FFFF BB04$_H$ | EIBD217 | FFFF BB64$_H$ | EIBD241 | FFFF BBC4$_H$ |
| EIBD170 | FFFF BAA8$_H$ | EIBD194 | FFFF BB08$_H$ | EIBD218 | FFFF BB68$_H$ | EIBD242 | FFFF BBC8$_H$ |
| EIBD171 | FFFF BAAC$_H$ | EIBD195 | FFFF BB0C$_H$ | EIBD219 | FFFF BB6C$_H$ | EIBD243 | FFFF BBCC$_H$ |
| EIBD172 | FFFF BAB0$_H$ | EIBD196 | FFFF BB10$_H$ | EIBD220 | FFFF BB70$_H$ | EIBD244 | FFFF BBD0$_H$ |
| EIBD173 | FFFF BAB4$_H$ | EIBD197 | FFFF BB14$_H$ | EIBD221 | FFFF BB74$_H$ | EIBD245 | FFFF BBD4$_H$ |
| EIBD174 | FFFF BAB8$_H$ | EIBD198 | FFFF BB18$_H$ | EIBD222 | FFFF BB78$_H$ | Reserved | FFFF BBD8$_H$ |
| EIBD175 | FFFF BABC$_H$ | EIBD199 | FFFF BB1C$_H$ | EIBD223 | FFFF BB7C$_H$ | Reserved | FFFF BBDC$_H$ |
| EIBD176 | FFFF BAC0$_H$ | EIBD200 | FFFF BB20$_H$ | EIBD224 | FFFF BB80$_H$ | Reserved | FFFF BBE0$_H$ |
| EIBD177 | FFFF BAC4$_H$ | EIBD201 | FFFF BB24$_H$ | EIBD225 | FFFF BB84$_H$ | Reserved | FFFF BBE4$_H$ |
| EIBD178 | FFFF BAC8$_H$ | EIBD202 | FFFF BB28$_H$ | EIBD226 | FFFF BB88$_H$ | Reserved | FFFF BBE8$_H$ |
| EIBD179 | FFFF BACC$_H$ | EIBD203 | FFFF BB2C$_H$ | EIBD227 | FFFF BB8C$_H$ | Reserved | FFFF BBEC$_H$ |
| EIBD180 | FFFF BAD0$_H$ | EIBD204 | FFFF BB30$_H$ | EIBD228 | FFFF BB90$_H$ | Reserved | FFFF BBF0$_H$ |
| EIBD181 | FFFF BAD4$_H$ | EIBD205 | FFFF BB34$_H$ | EIBD229 | FFFF BB94$_H$ | Reserved | FFFF BBF4$_H$ |
| EIBD182 | FFFF BAD8$_H$ | EIBD206 | FFFF BB38$_H$ | EIBD230 | FFFF BB98$_H$ | Reserved | FFFF BBF8$_H$ |
| EIBD183 | FFFF BADC$_H$ | EIBD207 | FFFF BB3C$_H$ | EIBD231 | FFFF BB9C$_H$ | Reserved | FFFF BBFC$_H$ |

### 6.4.9 SINTRn — Software Interrupt Registers (n = 0 to 2)

SINTR0 to SINTR2 are 8-bit registers to control software interrupts 0 to 2 (INTSW0 to INTSW2). Writing $01_H$ to these registers generates software interrupts 0 to 2 (INTSW0 to INTSW2). Writing $00_H$ during the generated interrupt handler clears the interrupt source. When these registers are read, the current register value is read. SINTR0 to SINTR2 are initialized to $00_H$ by a power-on reset.

**Access:** This register can be Read/written in 8/1-bit units.

**Address:** SINTR0: FFCD $E000_H$, SINTR1: FFCD $E004_H$, SINTR2: FFCD $E008_H$

**Value after reset:** $00_H$ This register is initialized by any reset.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SINTCn |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 6.16 SINTRn Register Contents (n = 0 to 2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SINTCn | Software Interrupt Request<br>This bit generates a software interrupt.<br>[Read operation]<br>The current bit value is read.<br>[Write operation]<br>Writing 1: Generates an interrupt.<br>Writing 1 while the SINTC bit is 1 is prohibited.<br>Writing 0: Clears the interrupt source. |

### 6.4.10 IPIR_CHn — Inter Processor Interrupt Register

Registers (IPIR_CH#) for interrupt communication between PEs are provided for two channels.

IPIR_CH1 and IPIR_CH2 are assigned to INTC1. An interrupt for specific PEs (including own PE) can be requested by manipulating bits corresponding to respective PEs.

This type of register is located in the CPU specific peripheral of each PE.

**Access:** This register can be Read/written in 32-, 16-, 8- or 1-bit units.

**Address:** IPIR_CH1: FFFE EC84$_H$, IPIR_CH2: FFFE EC88$_H$

**Value after reset:** 0000 0000$_H$
This register is initialized by any reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CHnPE2 | CHnPE1 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 6.17 IPIR_CHn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 30 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | CHnPE2 | Inter-PE Interrupt Request to PE2<br>Writing 1 to this bit generates an interrupt request to PE2. This bit is automatically cleared to 0 when the interrupt request has been notified.<br>0: Inter-PE interrupt request output is not specified or an interrupt request is not being output.<br>1: Interrupt request output is specified or an interrupt request is being output. |
| 0 | CHnPE1 | Inter-PE Interrupt Request to PE1<br>Writing 1 to this bit generates an interrupt request to PE1. This bit is automatically cleared to 0 when the interrupt request has been notified.<br>0: Inter-PE interrupt request output is not specified or an interrupt request is not being output.<br>1: Interrupt request output is specified or an interrupt request is being output. |

## 6.5 Interrupt Function System Registers

See **Section 3.2.1.2(3), Interrupt Function Registers**.

### 6.5.1 ISPR — Priority of interrupt being serviced

See **Section 3.2.1.2(3)(b), ISPR — Priority of interrupt being serviced register**.

### 6.5.2 PMR — Interrupt priority masking

See **Section 3.2.1.2(3)(c), PMR — Interrupt priority masking register**.

### 6.5.3 ICSR — Interrupt control status

See **Section 3.2.1.2(3)(d), ICSR — Interrupt control status register**.

### 6.5.4 INTCFG — Interrupt function setting

See **Section 3.2.1.2(3)(e), INTCFG — Interrupt function setting register**.

## 6.6 Operation When Acknowledging an Interrupt Exception

Check whether each exception that is reported during instruction execution is acknowledged according to the priority. The procedure for acknowledgment operation of each interrupt is shown below.

(1) Check whether the acknowledgment conditions are satisfied and whether exceptions are acknowledged according to their priority.

(2) Calculate the exception handler address according to the current PSW value*1.

(3) For FE-level non-maskable/maskable interrupts, the following processing is performed:

- Save the PC to the FEPC.
- Save the PSW to the FEPSW.
- Store the exception code in the FEIC.
- Update the PSW and MCTL.
- Store the exception handler address calculated in (2) in the PC, and then pass its control to the exception handler.

(4) For EI level exceptions, the following processing is performed:

- Save the PC to the EIPC.
- Save the PSW to the EIPSW.
- Store the exception code in the EIC.
- Update the PSW and MCTL.
- Store the exception handler address calculated in (2) in the PC, and then pass its control to the exception handler.

**Note 1.** For details, see **Section 6.9, Exception Handler Address**.

The following figure shows steps (1) to (4).

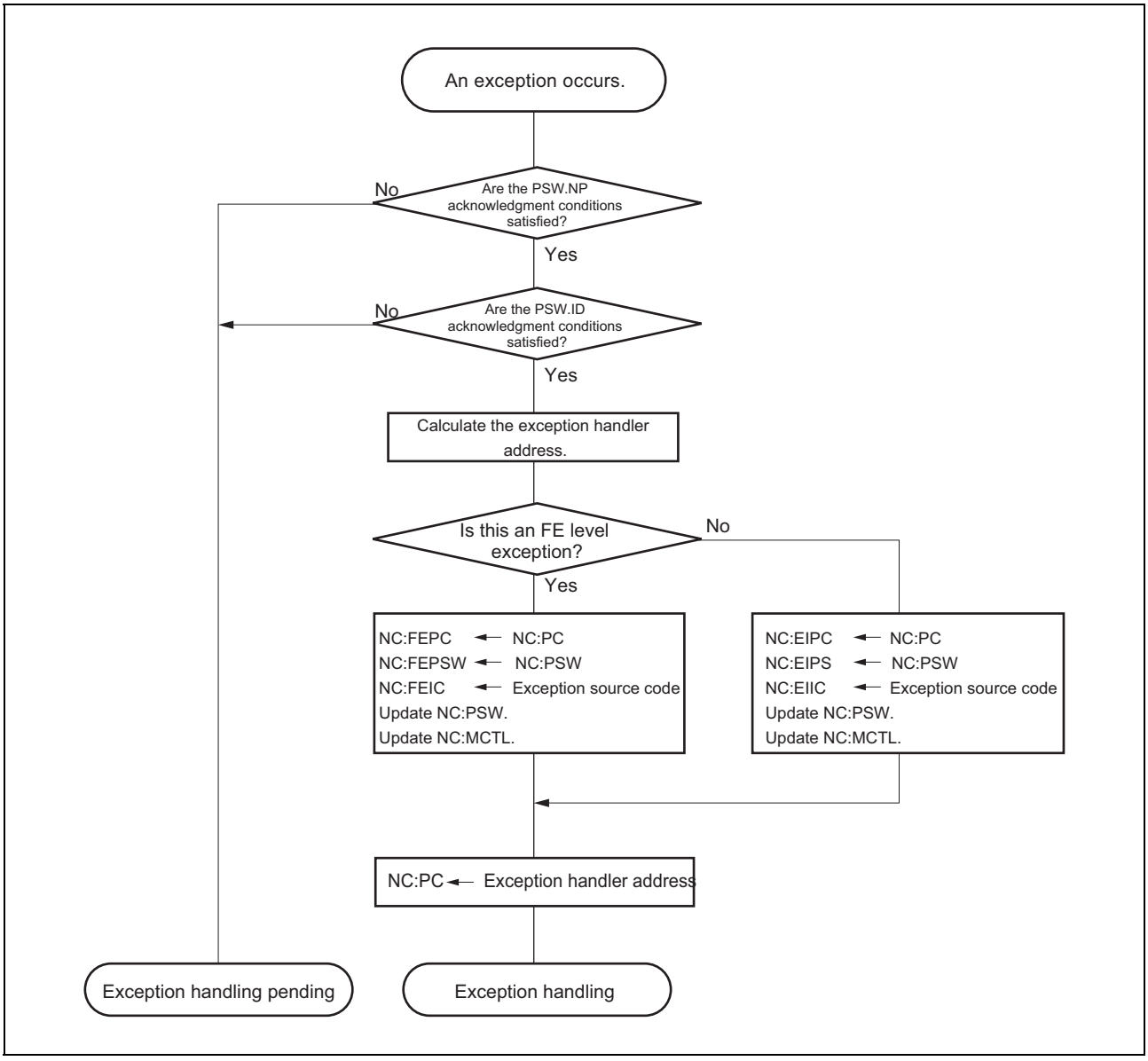



**Figure 6.2 Processing upon occurrence of interrupt request**

## 6.7 Return from Interrupts

To return from interrupt handling, execute the return instruction (EIRET or FERET) corresponding to each relevant interrupt level.

When a context has been saved in a stack and the like, the context must be restored before executing the return instruction.

When execution is returned from an irrecoverable interrupt or exception, the status before the exception occurs in the original program cannot be restored. Consequently, the execution result might differ from that when the interrupt does not occur.

The EIRET instruction is used to return from the EI-level maskable interrupt handling and the FERET instruction is used to return from FE-level maskable interrupt handling.

When the EIRET or FERET instruction is executed, the CPU performs the following processing and then passes its control to the return PC address:

(1) Return PC and PSW are loaded from the EIPC and EIPSW registers or the FEPC and FEPSW registers.

(2) Control is passed to the addresses indicated by the return PC and PSW that were loaded.

(3) When EIRET and EP = 0 and INTCFG.ISPC = 0, the CPU updates the ISPR register.

The flows for returning from exception handling using the EIRET and FERET instructions are shown below.

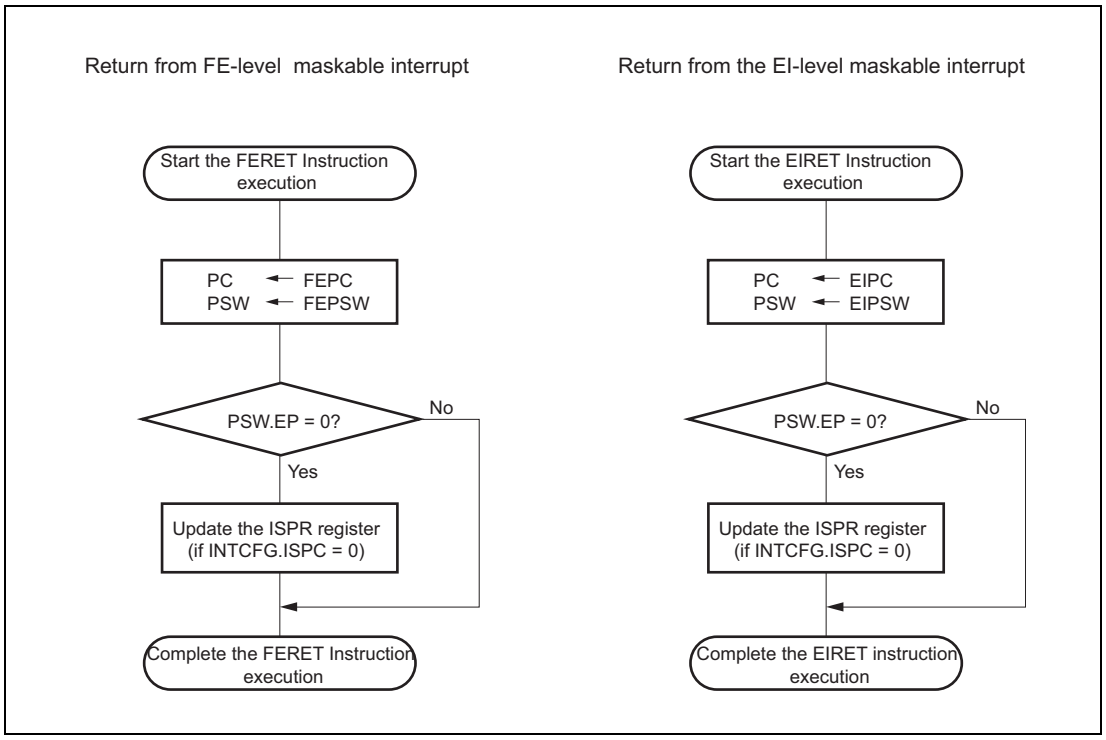



**Figure 6.3 Flow of Return from Interrupts**

## 6.8 Interrupt Operation

### 6.8.1 Mask function of EI level maskable interrupt INTn

Interrupt masking can be specified for each respective interrupt channel of INTn. Interrupt masking is performed by the following register settings.

**Table 6.18 Operation of the MKn Bit**

| EICn.MKn | Operation |
|---|---|
| 1 | Masks interrupt |
| 0 | Enables interrupt |

The EICn.MKn bits can also be read and written via the corresponding MKn bits of the IMRm registers. The interrupt mask state is reflected in both the EICn registers and the IMRm registers.

**Operation example**

(1) When a 1 is written to a IMRm.EIMKn bit, interrupts are prohibited for the corresponding channel.

(2) When the corresponding EICn.MKn bit is read, a 1 is returned.

**CAUTION**

**If the MKn bit is set to 0 while an interrupt request is withheld (RFn = 1), the interrupt service routine will be executed at that time (subject to the rules of interrupt prioritization). Even if an interrupt request is issued in software by setting the RFn bit to 1, the interrupt will not occur as long as the interrupt is masked with MKn = 1.**

**To remove an interrupt request that is withheld, clear the corresponding RFn bit in software (as would be done when polling for interrupt requests).**

**In case of using the external pin interrupt, it is necessary to clear the corresponding MKn bit after switched to the external pin by setting pin function registers.**

### 6.8.2 Interrupt priority level judgment

When FE level non-maskable interrupts (FENMI), FE level maskable interrupts (FEINT), and EI level maskable interrupts (INT) are input, priorities including other exceptions are determined, and the exception with the highest priority (including interrupts) is requested. Exceptions requested at the same time (including interrupts) are processed in a pre-allocated priority order (the default priority order). The priority orders of FENMI, FEINT, and INT interrupts are as follows.

FENMI > FEINT > EIINT

See **Table 6.5, RH850/P1x-C EI level maskable interrupt sources** and *the RH850G3M User's Manual: Software* for other exceptions.

For EIINTn interrupts, the priority level can be set independently for each interrupt source. Specify the priority level with the bits P3n to P0n. The interrupt priority levels can be set from 0 to 15: 0 is the

highest and 15 the lowest. Among multiple EIINTn interrupts with the same priority level, the interrupt with the lowest interrupt channel number has priority.

**Table 6.19 Example of EIINTn interrupt priority level settings and priority levels**

| EIINTn | EICn.P[3:0]n setting | Priority level during operation |
|---|---|---|
| EIINT0 | 3 | 10 |
| EIINT1 | 4 | 11 |
| EIINT2 | 0 | 1 |
| EIINT3 | 0 | 2 |
| EIINT4 | 1 | 3 |
| EIINT5 | 2 | 6 |
| EIINT6 | 2 | 7 |
| EIINT7 | 1 | 4 |
| EIINT8 | 1 | 5 |
| EIINT9 | 2 | 8 |
| EIINT10 | 2 | 9 |

The interrupt controller executes multiple interrupt handling when another interrupt request is acknowledged while an interrupt processing has been executed. When multiple EIINTn interrupts are requested at the same time, the interrupt to be acknowledged is determined by the following procedure.

#### 6.8.2.1 Comparison with the priority level as the interrupt currently being serviced

Interrupts with the same or lower priority level as the interrupt currently being serviced are held pending.

The priority level of the interrupt currently being serviced is shown in the ISPR register.

Interrupts with a higher priority level than the interrupt currently being serviced proceed to the next priority judgment stage.

#### 6.8.2.2 Masking through priority mask register (PMR)

Only interrupts enabled by the PMR register proceed to the next priority judgment stage.

For the PMR register, see the **Section 3.2.1.2(3)(c), PMR — Interrupt priority masking register** or *the RH850G3M User's Manual: Software*.

#### 6.8.2.3 The requested interrupt source with the highest priority level is selected

When interrupts are being simultaneously requested from multiple sources, the interrupt source from the highest priority level, with the smallest interrupt channel number is selected.

#### 6.8.2.4 Interrupt hold by CPU

Interrupt acknowledgment is pending according to the state of the NP and ID bits of the PSW register.

At this time, priority judgment among EIINTn interrupts, and priority judgment among EIINTn, FEINT and FENMI interrupts is performed even while interrupt acknowledgment is pending, and the interrupt with the highest priority is selected upon realization of the acknowledgment condition.

Example

An EIINTn interrupt with the priority level 5 has already been requested and interrupt generation is pending because the value of the PSW.ID bit is 1. If a subsequent EIINTn interrupt with the priority

level 3 is requested and the PSW.ID bit is cleared to 0, the latter EIINT interrupt (with the priority level 3) will be generated.

**Figure 6.4** shows an example of multiple interrupt handling when another interrupt request is acknowledged while an interrupt processing has been executed.

When an interrupt request signal is acknowledged, the PSW.ID flag is automatically set to 1. Therefore, the ID flag should be cleared to 0 to execute multiple interrupt handling. Specifically, execute the EI instruction and the like in an interrupt handling program to enable the interrupt.

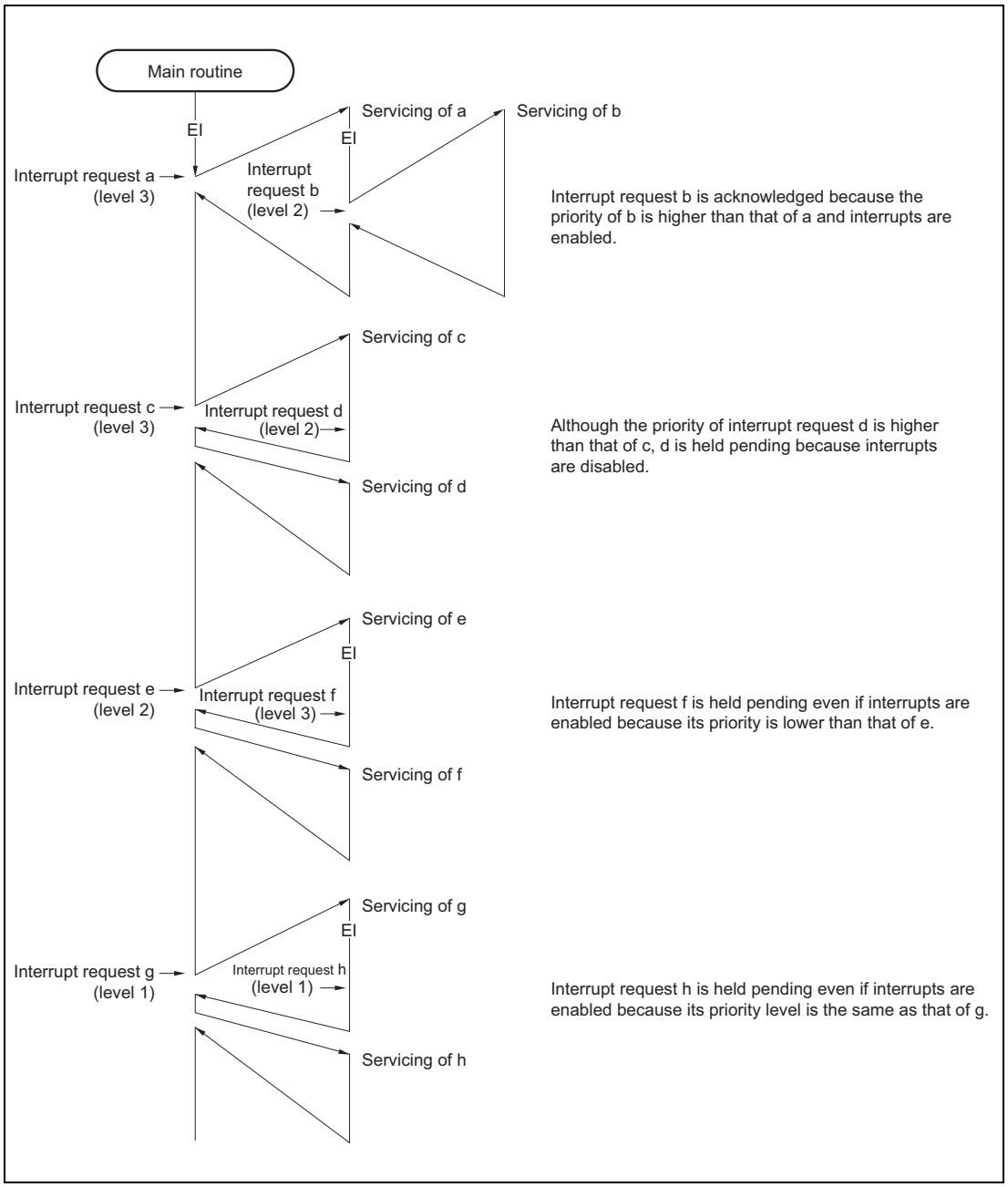
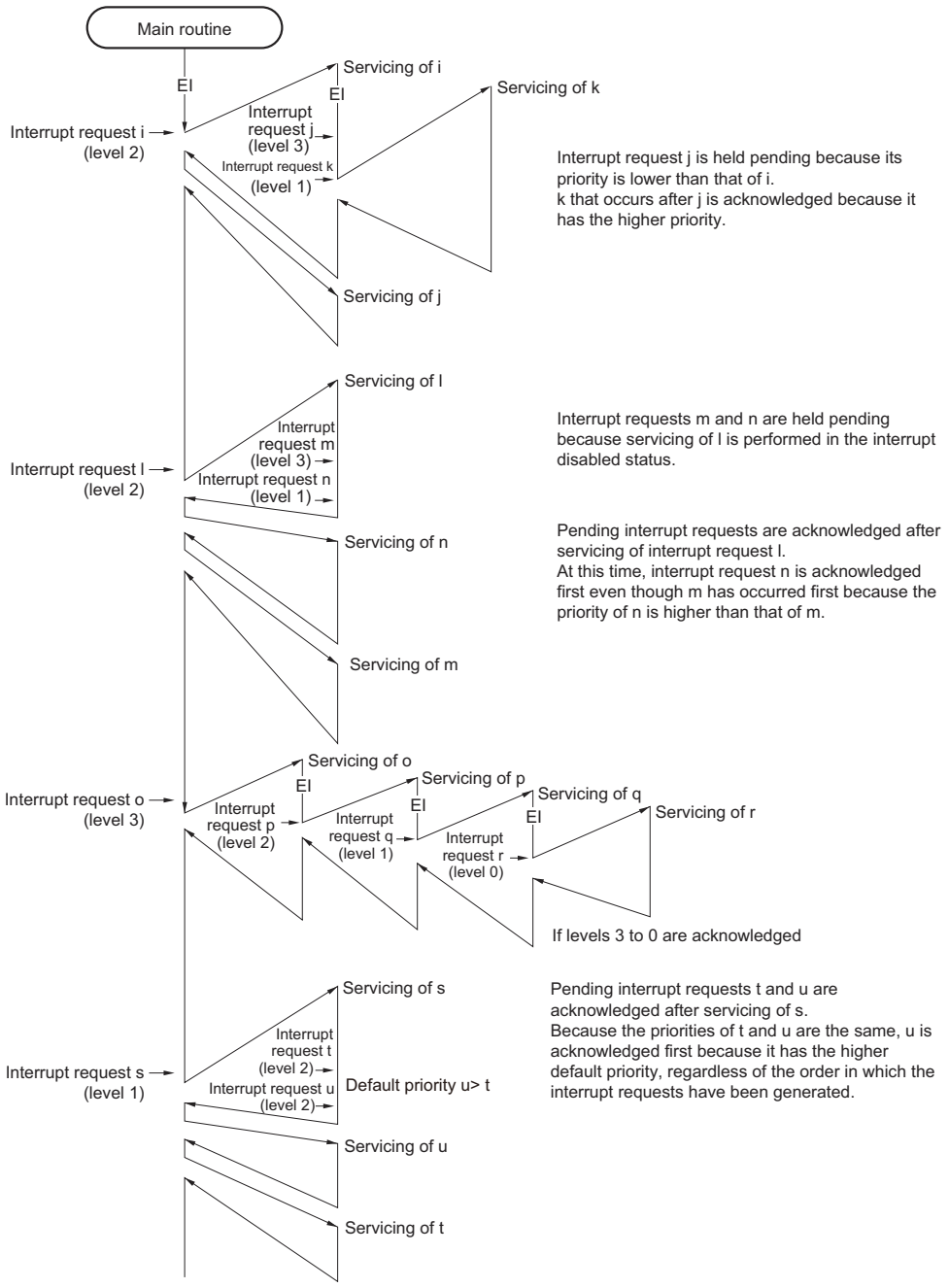



**Figure 6.4 Example of processing in which an interrupt request is issued while another interrupt is being serviced (1)**



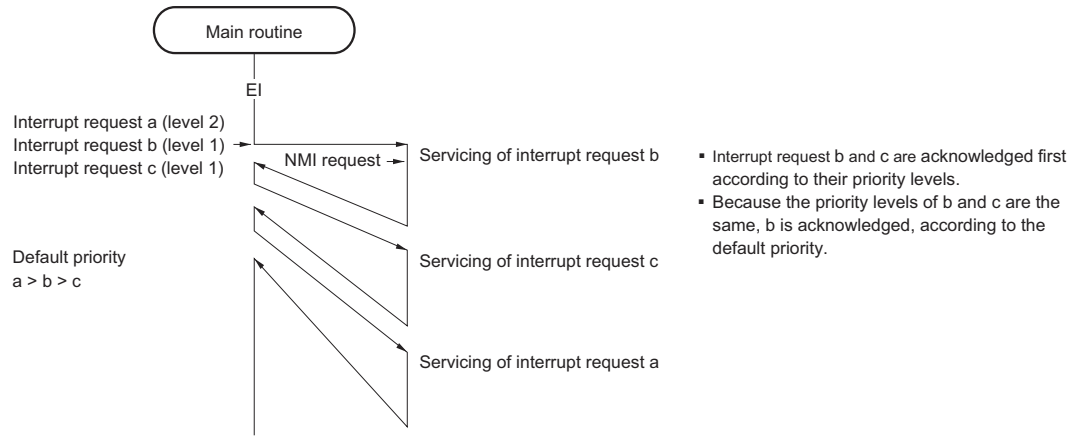

Note 1. “a” to “u” in the figure are the temporary names of interrupt request signals shown for the sake of explanation.

Note 2. The default priority in the figure indicates the relative priority between two interrupt request signals.

**Figure 6.5 Example of processing in which an interrupt request signal is issued while another interrupt is being serviced (2)**

**CAUTION**

**To perform multiple interrupt servicing, the values of the EIPC and EIPSW registers must be saved before executing the EI instruction. When returning from multiple interrupt servicing, restore the values of EIPC and EIPSW after executing the DI instruction.**




Note 1. “a” to “c” in the figure are the temporary names of interrupt request signals shown for the sake of explanation.

Note 2. The default priority in the figure indicates the relative priority between two interrupt request signals.

**Figure 6.6 Example of servicing simultaneously generated interrupt requests**

### 6.8.3 Interrupt request acceptance condition and the priority

See *RH850G3M User's Manual: Software*.

### 6.8.4 Exception priority of interrupts and the priority mask

See *RH850G3M User's Manual: Software*.

### 6.8.5 Interrupt priority mask

See *RH850G3M User's Manual: Software*.

### 6.8.6 Priority mask function

The priority mask function prohibits in batch EIINTn interrupts of the specified interrupt priority levels.

The interrupt priority levels to be masked are specified in the PMR register. Masking and acknowledgment can be set for each priority level.

The following operations are possible using this function:

- Temporary prohibition of interrupts that have a priority level that is lower than a given priority level
- Temporary prohibition of interrupts that have a given priority level

**Table 6.20 Priority mask function**

| PMR.PMm | Operation |
|---|---|
| 0 | Acknowledges requests from priority level m interrupt source. |
| 1 | Masks requests from priority level m interrupt source. |

**Note:** m = 0 to 15

The PMR register prohibits vectoring to the interrupt service when an interrupt request of the specified priority is generated.

The presence of EIINTn interrupts held pending with this function can be checked with **Section 6.8.8, Interrupt management**.

For details on the PMR register, see **Section 3.2.1.2(3)(c), PMR — Interrupt priority masking register**, or *the RH850G3M User's Manual: Software*.

### 6.8.7 FEINT Source selection

There are three sources for the FEINT: ECM and STM interrupt and terminal NMI.

These three sources are mapped to the interrupt controller INTC1. There is a status flag register (See **Section 6.4.5, FEINTFn — FE level maskable interrupt event status register**), an event mask register (See **Section 6.4.7, FEINTFMSKn — FE level maskable interrupt event mask register**) and an event clear register (See **Section 6.4.6, FEINTFCn — FEINT event clear register**) for each source. The event status flag has to be cleared by software. If an event is masked (FEINTEMSKn = 1) the corresponding bit FEINTESTATn is not set.

The event status flag should be cleared by software in the beginning of the interrupt service routine. If there is a pending interrupt flag after leaving the interrupt service routine the interrupt will be asserted again until all flags in FEINTESTAT are cleared.

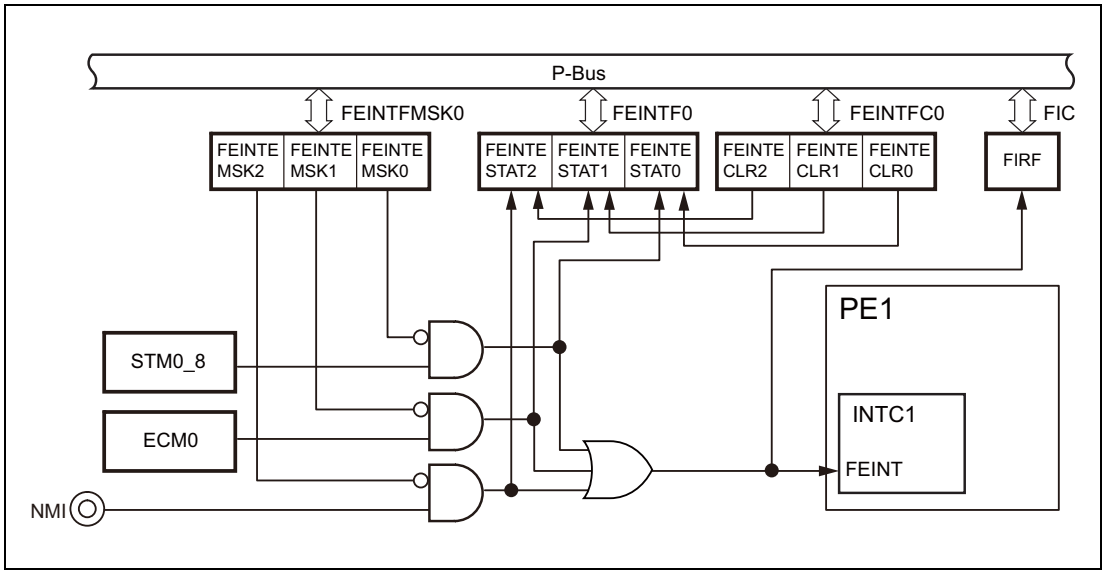



**Figure 6.7 FEINT Source selection for PE1**

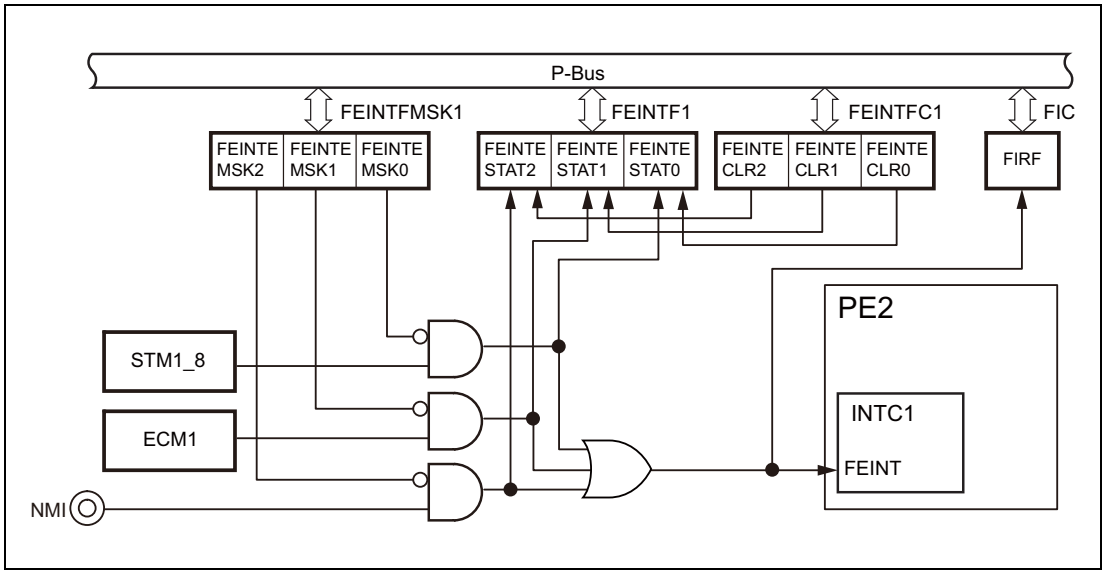



**Figure 6.8 FEINT Source selection for PE2**

### 6.8.8 Interrupt management

The suspended interrupt can be checked in the RH850/P1x-C. For details, see *the RH850G3M User's Manual: Software*.

## 6.9 Exception Handler Address

For the RH850/P1x-C, the exception handler address used for execution during reset input, exception acknowledgment, or interrupt acknowledgment can be changed according to the settings.

The exception handler address for reset and exception (including interrupt) is determined with the direct vector method, in which the reference point of the exception handler address can be changed by using the PSW.EBV bit, the RBASE register, and the EBASE register. For interrupts, the direct vector method and table reference method can be selected for each channel.

If the table reference method is selected, execution can branch to the address indicated by the exception handler table allocated in the memory.

**CAUTION**

**The exception handler address of EINTn selected using the direct vector method differs from that of the V850E2 core products. In the V850E2 core products, a different exception handler address is individually assigned to each interrupt channel (EINTn). In the RH850/P1x-C, one exception handler address is assigned to each interrupt priority. Consequently, interrupts that have the same priority level branch to the same exception handler.**

### 6.9.1 Direct Vector Method

The CPU uses the result of adding the offset shown in **Table 6.21, Selection of Base Register/ Offset Address** to the base address indicated by the RBASE or EBASE register as the exception handler address.

Select whether the RBASE or EBASE register is used as the base address in the PSW.EBV bit. When the PSW.EBV bit is set to 1, the value of the EBASE register is used as the base address. When the PSW.EBV bit is cleared to 0, the value of the RBASE register is used as the base address.

For reset input , however, the RBASE register is always used for reference.

In addition, user interrupts refer to the RINT bit of the corresponding base register, and reduce the offset address according to the bit status. If the RBASE.RINT bit or EBASE.RINT bit is set to 1, all user interrupts are handled using an offset of $100_H$. If the bit is cleared to 0, the offset address is determined according to **Table 6.21, Selection of Base Register/Offset Address**.

**Note 1.** Exception acknowledgment itself may sometimes update the status of the PSW.EBV bit. In this case, the base register is selected based on the new bit value.

**Note 2.** The exceptions that always refer to the RBASE register are determined according to the hardware specifications.

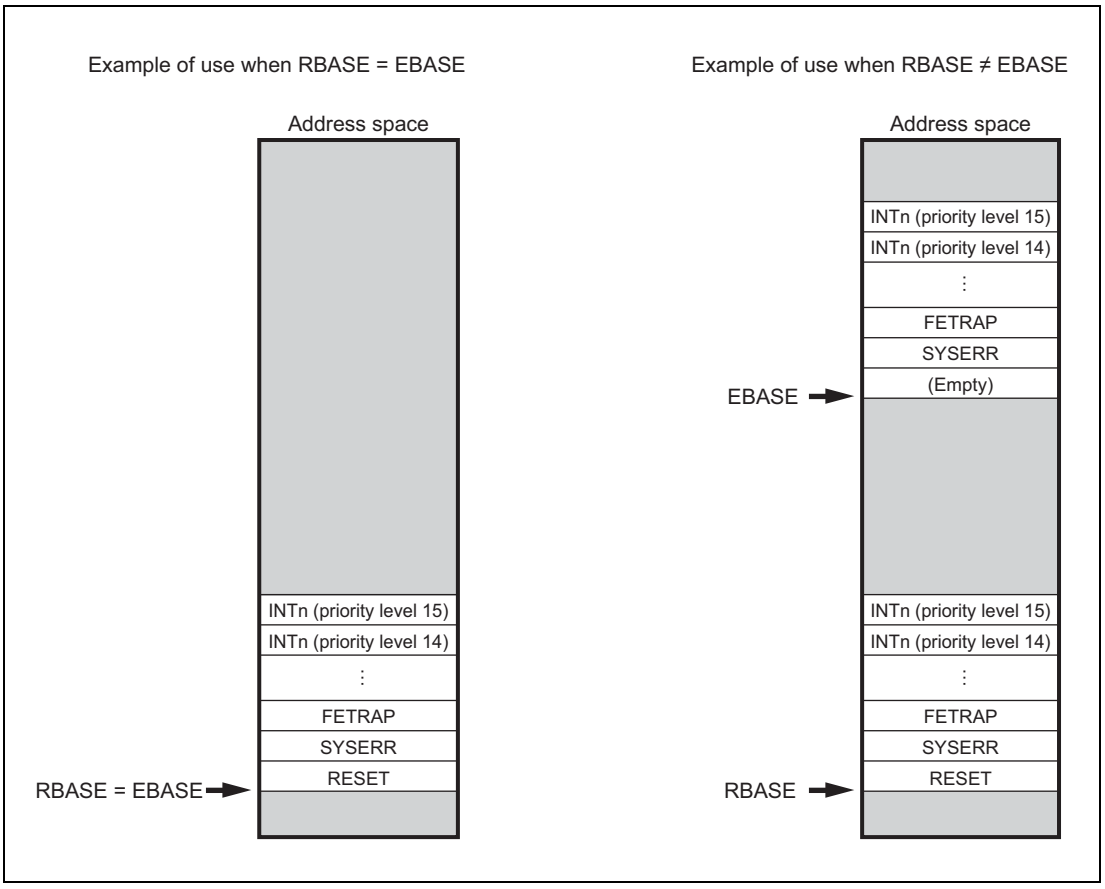



**Figure 6.9 Direct Vector Method**

The table below shows how base register selection and offset address reduction function for each exception to determine the exception handler address. The value of the PSW bit determines the exception handler on the basis of the value after updated by the acknowledgment of an exception.

**Table 6.21 Selection of Base Register/Offset Address**

| | PSW.EBV = 0 | PSW.EBV = 1 | RINT = 0 | RINT = 1 |
|---|---|---|---|---|
| **Exception/Interrupt** | **Base register** | | **Offset address** | |
| RESET | RBASE | N.A. | $000_H$ | $000_H$ |
| SYSERR | | EBASE | $010_H$ | $010_H$ |
| FETRAP | | | $030_H$ | $030_H$ |
| TRAP0 | | | $040_H$ | $040_H$ |
| TRAP1 | | | $050_H$ | $050_H$ |
| RIE | | | $060_H$ | $060_H$ |
| FPP/FPI | | | $070_H$ | $070_H$ |
| UCPOP | | | $080_H$ | $080_H$ |
| MIP/MDP/ITLBE/DTLBE | | | $090_H$ | $090_H$ |
| PIE | | | $0A0_H$ | $0A0_H$ |
| Debug | | | $0B0_H$ | $0B0_H$ |
| MAE | | | $0C0_H$ | $0C0_H$ |
| Reserved | | | $0D0_H$ | $0D0_H$ |
| FENMI | | | $0E0_H$ | $0E0_H$ |
| FEINT | | | $0F0_H$ | $0F0_H$ |
| EIINTn (Priority level 0) | | | $100_H$ | $100_H$ |
| EIINTn (Priority level 1) | | | $110_H$ | |
| EIINTn (Priority level 2) | | | $120_H$ | |
| EIINTn (Priority level 3) | | | $130_H$ | |
| EIINTn (Priority level 4) | | | $140_H$ | |
| EIINTn (Priority level 5) | | | $150_H$ | |
| EIINTn (Priority level 6) | | | $160_H$ | |
| EIINTn (Priority level 7) | | | $170_H$ | |
| EIINTn (Priority level 8) | | | $180_H$ | |
| EIINTn (Priority level 9) | | | $190_H$ | |
| EIINTn (Priority level 10) | | | $1A0_H$ | |
| EIINTn (Priority level 11) | | | $1B0_H$ | |
| EIINTn (Priority level 12) | | | $1C0_H$ | |
| EIINTn (Priority level 13) | | | $1D0_H$ | |
| EIINTn (Priority level 14) | | | $1E0_H$ | |
| EIINTn (Priority level 15) | | | $1F0_H$ | |

Base register selection is used to execute the exception handling for reset and some hardware errors by using the programs in a relatively reliable area such as ROM instead of the areas that are easily affected by software errors such as RAM and cache area. The user interrupt offset address reduction function is used to reduce the memory occupation size required by the exception handler for specific system-internal operating modes. The main purpose of this is to minimize the amount of memory consumed in operating modes that use only the minimum functionality, for example, during system maintenance and diagnosis.

### 6.9.2 Table Reference Method

In the Direct Vector Method, there is one user-interrupt exception handler for each interrupt priority level, and interrupt channels that indicate multiple interrupts with the same priority branch to the same interrupt handler, but some users might want to use different code areas for each interrupt handler from the beginning.

The RH850/P1x-C defines the table reference method for interrupts that assume the above usage.

For the table reference method, if the table reference method is specified as the interrupt channel vector selection method in the interrupt controller and the like, the method for determining the exception handler address when an interrupt request corresponding to that interrupt channel is acknowledged differs as follows.

<1> In any of the following cases, the exception handler address is determined by using the direct vector method:

- When PSW.EBV = 0 and RBASE.RINT = 1
- When PSW.EBV = 1 and EBASE.RINT = 1
- When the interrupt channel setting is not the table reference method

<2> In cases other than <1>, calculate the table reference position.
Exception handler address read position = INTBP register + channel number × 4 bytes

<3> Read word data starting at the interrupt handler address read position calculated in <2>.

<4> Use the word data read in <3> as the exception handler address.

**Table 6.22** shows the exception handler address read positions corresponding to each interrupt channel and **Figure 6.10** shows an overview of the placement in memory.

**Table 6.22 Exception Handler Address Expansion**

| Type of Interrupt | Exception Handler Address Read Position |
|---|---|
| EI level maskable interrupt channel 0 | INTBP register value + 0 × 4 |
| EI level maskable interrupt channel 1 | INTBP register value + 1 × 4 |
| EI level maskable interrupt channel 2 | INTBP register value + 2 × 4 |
| : | : |
| EI level maskable interrupt channel 254 | INTBP register value + 254 × 4 |
| EI level maskable interrupt channel 255 | INTBP register value + 255 × 4 |

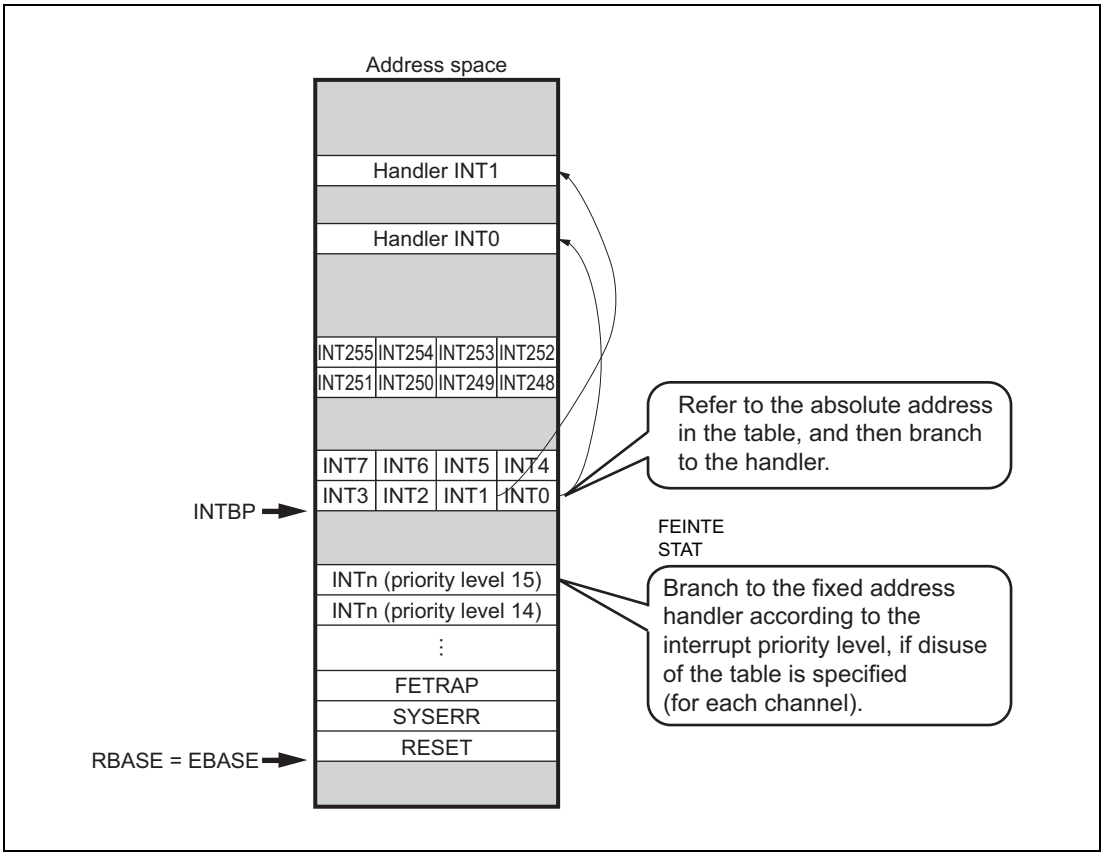



**Figure 6.10 Overview of the Table Reference Method**

## 6.10 Broadcast notification function processor elements

Each interrupt has its own interrupt control register and interrupt bind register. Each interrupt can be assigned to one dedicated PE by setting the corresponding EIBD.CST bit to ‘0’ (value after reset) and the EIBDn.PEID[2:0] to the number of the target PE. If an interrupt should be handled by all processor elements the broadcast function is used. In this case the bit EIBDn.CST has to be set to ‘1’.One of the 4 available broadcast channels can be selected by programming the EIBDn accordingly. There Interrupt inputs EIINT[7:4] of each INTC1 are used for the broadcast interrupt handling. These are no dedicated broadcast channels (this means the assignment to the interrupt inputs of the INTC1 has to done during specification phase and will be implemented in hardware. Thus the assignment of the broadcast channels (outputs of INTC2) to the INTC1 inputs cannot be changed by software).

- It is prohibited to allocate two or more interrupt channels to the same broadcast notification port of INTC1.
- If an interrupt EIINTn is assigned to a broadcast notification channel the interrupt priority information in EICn.EIPm is ignored. Thus the interrupt vector of the assigned broadcast notification interrupt is used.
- It is possible to mask a broadcast notification interrupt for a dedicated PEx by setting the mask bit EICx.IMK in the corresponding INTC1 macro.
- The interrupt vector for a broadcast notification interrupt is the vector of the corresponding EIINT interrupt input at the INTC1 macro (broadcast channel).

### 6.10.1 Example:

- Peripheral interrupt INTRXD is assigned to INTC2, channel 100,
- Broadcast channels 3 to 0 are used.
- The broadcast channels are connected to INTC1.eiint[7:4]
- EIBD100.CST is set to ‘1’ (broadcast notification enabled)
- EIBD100.BCP[2:0] I set to 0x2 (broadcast notification channel 2 is selected)
- Thus INTRXD interrupt is assigned to PE1 and PE2, interrupt channel 6
- The priority of this interrupt is now defined by EIC6.EIP[3:0] (for each PE separately)
- The corresponding mask bit is EIC6.IMK and the interrupt request flag is EIC6.IRF (thus the interrupt can be masked/enable for each PE separately in its own INTC1)

In case of an interrupt from the peripheral the INTRXD signal would be active. This interrupt request is provided from INTC2 to INTC1 via the broadcast notification interrupt INTBN2. Thus the EIC6.RF is set for both INTC1 macros. Both PE’s will execute the corresponding interrupt service routine and each EIC6.RF flags are cleared automatically.

Corresponding register setting:

**Table 6.23 Corresponding register setting (INTC2)**

| Register | Bit | Content | Description |
|---|---|---|---|
| EIC100 | ICT | 1/0 | User readable |
| | IRF | Don't care | No function if broadcast notification is set |
| | IMK | 0 | Set to 0 during broadcast interrupt service |
| | ITB | Don't care | No function if broadcast notification is set |
| | IP | Don't care | No function if broadcast notification is set (priority is defined by BN2) |
| EIBD100 | CST | 1 | Broadcast notification is enabled |
| | BCP | 010 | Broadcast notification channel = 2 |
| | PEID | Don't care | No function if broadcast notification is set |

**Table 6.24 Corresponding register setting (INTC1)**

| Register | Bit | Content | Description |
|---|---|---|---|
| EIC6 | ICT | 1/0 | User readable |
| | IRF | 1/0 | Interrupt flag of the broadcast notification channel 2 |
| | IMK | 1/0 | Interrupt mask bit of the broadcast notification channel 2 |
| | ITB | 1/0 | User programmable |
| | IP | 0 to15 | User programmable |
| EIBD6 | CST | 0 | Fixed (cannot be changed) |
| | BCP | 000 | Fixed (cannot be changed) |
| | INTC1.PEID | ID (PE1)<br>ID (PE2) | Fixed (cannot be changed) |

## 6.11 Interrupt latency

The interrupt latency is summarized in **Table 6.25, Interrupt latency**. The given values describe the response time from the assertion of the interrupt trigger at the INTC module and the execution of the first instruction inside the interrupt handling routine. The response time is the typical time for the given scenario. However, it is assumed that no other bus master accesses the code flash in case of an instruction cache miss. The unit of the response time is PE clock cycle.

**Table 6.25 Interrupt latency**

| Device | Instruction cache | Interrupt detection | Exception handler addressing method | Address reference table location | Latency (PE1/2) | | Latency (ICUMC) | |
|---|---|---|---|---|---|---|---|---|
| | | | | | INTC1 | INTC2 | INTC1 | INTC2 |
| P1M-C, P1H-C and P1H-CE<br>240 MHz | hit | edge | direct vector | NA | 8 | 16 | 18 | 24 |
| | miss | edge | direct vector | NA | 13 | 21 | 26 | 32 |
| | hit | level | direct vector | NA | 7 | 13 | 15 | 21 |
| | miss | level | direct vector | NA | 12 | 18 | 23 | 29 |
| | hit | edge | table reference | Local RAM | 12 | 20 | 27 | 33 |
| | miss | edge | table reference | Local RAM | 17 | 25 | 35 | 41 |
| | hit | level | table reference | Local RAM | 11 | 17 | 24 | 30 |
| | miss | level | table reference | Local RAM | 16 | 22 | 32 | 38 |
| | hit | edge | table reference | Code flash | 17 | 25 | 39 | 45 |
| | miss | edge | table reference | Code flash | 22 | 30 | 47 | 53 |
| | hit | level | table reference | Code flash | 16 | 22 | 36 | 42 |
| | miss | level | table reference | Code flash | 21 | 27 | 44 | 50 |

## 6.12 Difference among P1M-C, P1H-C and P1H-CE

Table 6.26 Difference among P1M-C, P1H-C and P1H-CE (1/7)

| Level | Symbol | Source | Generating Condition or Factor | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|---|
| FE | FENMI | WDT | Secure WDT overflow interrupt | √ | √ | √ | √ |
| FE | FEINT*2 | ECM<br>STM<br>Pin | non mask-able interrupt from ECM,<br>non mask-able interrupt (STM interrupt 8 for STM0),<br>non mask-able interrupt for Terminal NMI interrupt | √ | √ | √ | √ |
| EI | EIINT0 | ECM | mask-able interrupt from ECM | √ | √ | √ | √ |
| EI | EIINT1 | IPIR | IPIR_CH1 (between PE interrupt) | — | √ | √ | √ |
| EI | EIINT2 | IPIR | IPIR_CH2 (between PE interrupt) | — | √ | √ | √ |
| EI | EIINT3 | ICUMC | interrupt from ICUMC | √ | √ | √ | √ |
| EI | EIINT4 | INTC2 | broadcast notification 0 | √ | √ | √ | √ |
| EI | EIINT5 | INTC2 | broadcast notification 1 | √ | √ | √ | √ |
| EI | EIINT6 | INTC2 | broadcast notification 2 | √ | √ | √ | √ |
| EI | EIINT7 | INTC2 | broadcast notification 3 | √ | √ | √ | √ |
| EI | EIINT8 | WDTA | Interval timer interrupt (75% interrupt) | √ | √ | √ | √ |
| EI | EIINT9 | SW | Software interrupt0 | √ | √ | √ | √ |
| EI | EIINT10 | SW | Software interrupt1 | √ | √ | √ | √ |
| EI | EIINT11 | SW | Software interrupt2 | √ | √ | √ | √ |
| EI | EIINT12 | STM | STM interrupt 2 (STM0) | √ | √ | √ | √ |
| EI | EIINT13 | STM | STM interrupt 3 (STM0) | √ | √ | √ | √ |
| EI | EIINT14 | STM | STM interrupt 4 (STM0) | √ | √ | √ | √ |
| EI | EIINT15 | STM | STM interrupt 5 (STM0) | √ | √ | √ | √ |
| EI | EIINT16 | DMA | DMA transfer completion for ch0 /<br>DMA count match interrupt for ch0 | √ | √ | √ | √ |
| EI | EIINT17 | DMA | DMA transfer completion for ch1 /<br>DMA count match interrupt for ch1 | √ | √ | √ | √ |
| EI | EIINT18 | DMA | DMA transfer completion for ch2 /<br>DMA count match interrupt for ch2 | √ | √ | √ | √ |
| EI | EIINT19 | DMA | DMA transfer completion for ch3 /<br>DMA count match interrupt for ch3 | √ | √ | √ | √ |
| EI | EIINT20 | DMA | DMA transfer completion for ch4 /<br>DMA count match interrupt for ch4 | √ | √ | √ | √ |
| EI | EIINT21 | DMA | DMA transfer completion for ch5 /<br>DMA count match interrupt for ch5 | √ | √ | √ | √ |
| EI | EIINT22 | DMA | DMA transfer completion for ch6 /<br>DMA count match interrupt for ch6 | √ | √ | √ | √ |
| EI | EIINT23 | DMA | DMA transfer completion for ch7 /<br>DMA count match interrupt for ch7 | √ | √ | √ | √ |
| EI | EIINT24 | DMA | DMA transfer completion for ch8 /<br>DMA count match interrupt for ch8 | √ | √ | √ | √ |
| EI | EIINT25 | DMA | DMA transfer completion for ch9 /<br>DMA count match interrupt for ch9 | √ | √ | √ | √ |
| EI | EIINT26 | DMA | DMA transfer completion for ch10 /<br>DMA count match interrupt for ch10 | √ | √ | √ | √ |
| EI | EIINT27 | DMA | DMA transfer completion for ch11 /<br>DMA count match interrupt for ch11 | √ | √ | √ | √ |

Table 6.26 Difference among P1M-C, P1H-C and P1H-CE (2/7)

| Level | Symbol | Source | Generating Condition or Factor | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|---|
| EI | EIINT28 | DMA | DMA transfer completion for ch12 / DMA count match interrupt for ch12 | √ | √ | √ | √ |
| EI | EIINT29 | DMA | DMA transfer completion for ch13 / DMA count match interrupt for ch13 | √ | √ | √ | √ |
| EI | EIINT30 | DMA | DMA transfer completion for ch14 / DMA count match interrupt for ch14 | √ | √ | √ | √ |
| EI | EIINT31 | DMA | DMA transfer completion for ch15 / DMA count match interrupt for ch15 | √ | √ | √ | √ |
| EI | EIINT32 | Pin | External Interrupt0 | √ | √ | √ | √ |
| EI | EIINT33 | Pin | External Interrupt1 | √ | √ | √ | √ |
| EI | EIINT34 | Pin | External Interrupt2 | √ | √ | √ | √ |
| EI | EIINT35 | Pin | External Interrupt3 | √ | √ | √ | √ |
| EI | EIINT36 | Pin | External Interrupt4 | √ | √ | √ | √ |
| EI | EIINT37 | TEMPS | Temperature sensor error interrupt | √ | √ | √ | √ |
| EI | EIINT38 | TEMPS | temperature measurement end interrupt | √ | √ | √ | √ |
| EI | EIINT39 | TEMPS | Triggered if state machine change the stage by the temperature rising or falling in the guaranteed temperature range | √ | √ | √ | √ |
| EI | EIINT40 | NTU | NTU loss interrupt | √ | √ | √ | √ |
| EI | EIINT41 | STM | STM interrupt 0 (STM0) | √ | √ | √ | √ |
| EI | EIINT42 | STM | STM interrupt 1 (STM0) | √ | √ | √ | √ |
| EI | EIINT43 | STM | STM interrupt 0 (STM1) | — | √ | √ | √ |
| EI | EIINT44 | STM | STM interrupt 1 (STM1) | — | √ | √ | √ |
| EI | EIINT45 | STM | STM interrupt 2 (STM1) | — | √ | √ | √ |
| EI | EIINT46 | STM | STM interrupt 3 (STM1) | — | √ | √ | √ |
| EI | EIINT47 | GTM | GTM Error Interrupt | √ | √ | √ | √ |
| EI | EIINT48 | GTM | AEI Shared interrupt | √ | √ | √ | √ |
| EI | EIINT49 | GTM | ARU_NEW_DATA0 interrupt | √ | √ | √ | √ |
| EI | EIINT50 | GTM | ARU_NEW_DATA1 interrupt | √ | √ | √ | √ |
| EI | EIINT51 | GTM | ARU_ACC_ACK interrupt | √ | √ | √ | √ |
| EI | EIINT52 | GTM | CMP Shared interrupt | √ | √ | √ | √ |
| EI | EIINT53 | GTM | TIM Shared interrupts (TIM0_IRQ0) | √ | √ | √ | √ |
| EI | EIINT54 | GTM | TIM Shared interrupts (TIM0_IRQ2) | √ | √ | √ | √ |
| EI | EIINT55 | GTM | TIM Shared interrupts (TIM0_IRQ4) | √ | √ | √ | √ |
| EI | EIINT56 | GTM | TIM Shared interrupts (TIM0_IRQ6) | √ | √ | √ | √ |
| EI | EIINT57 | GTM | TIM Shared interrupts (TIM1_IRQ0) | √ | √ | √ | √ |
| EI | EIINT58 | GTM | TIM Shared interrupts (TIM1_IRQ2) | √ | √ | √ | √ |
| EI | EIINT59 | GTM | TIM Shared interrupts (TIM1_IRQ4) | √ | √ | √ | √ |
| EI | EIINT60 | GTM | TIM Shared interrupts (TIM1_IRQ6) | √ | √ | √ | √ |
| EI | EIINT61 | GTM | MCS Interrupt for channel (MCS0_IRQ0) | √ | √ | √ | √ |
| EI | EIINT62 | GTM | MCS Interrupt for channel (MCS0_IRQ2) | √ | √ | √ | √ |
| EI | EIINT63 | GTM | MCS Interrupt for channel (MCS0_IRQ4) | √ | √ | √ | √ |
| EI | EIINT64 | GTM | MCS Interrupt for channel (MCS0_IRQ6) | √ | √ | √ | √ |
| EI | EIINT65 | GTM | MCS Interrupt for channel (MCS0_IRQ8) | √ | √ | √ | √ |

Table 6.26 Difference among P1M-C, P1H-C and P1H-CE (3/7)

| Level | Symbol | Source | Generating Condition or Factor | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|---|
| EI | EIINT66 | GTM | MCS Interrupt for channel (MCS1_IRQ1) | — | √ | √ | √ |
| EI | EIINT67 | GTM | MCS Interrupt for channel (MCS1_IRQ3) | — | √ | √ | √ |
| EI | EIINT68 | GTM | MCS Interrupt for channel (MCS1_IRQ5) | — | √ | √ | √ |
| EI | EIINT69 | GTM | ATOM Shared interrupts (ATOM0_IRQ0) | √ | √ | √ | √ |
| EI | EIINT70 | GTM | ATOM Shared interrupts (ATOM0_IRQ2) | √ | √ | √ | √ |
| EI | EIINT71 | GTM | ATOM Shared interrupts (ATOM0_IRQ4) | √ | √ | √ | √ |
| EI | EIINT72 | GTM | ATOM Shared interrupts (ATOM0_IRQ6) | √ | √ | √ | √ |
| EI | EIINT73 | GTM | ATOM Shared interrupts (ATOM1_IRQ0) | √ | √ | √ | √ |
| EI | EIINT74 | GTM | ATOM Shared interrupts (ATOM1_IRQ2) | √ | √ | √ | √ |
| EI | EIINT75 | GTM | ATOM Shared interrupts (ATOM1_IRQ4) | — | √ | √ | √ |
| EI | EIINT76 | GTM | ATOM Shared interrupts (ATOM1_IRQ6) | — | √ | √ | √ |
| EI | EIINT77 | GTM | ATOM Shared interrupts (ATOM2_IRQ0) | — | √ | √ | √ |
| EI | EIINT78 | GTM | ATOM Shared interrupts (ATOM2_IRQ2) | — | √ | √ | √ |
| EI | EIINT79 | GTM | ATOM Shared interrupts (ATOM2_IRQ4) | — | √ | √ | √ |
| EI | EIINT80 | ADCF | Error Interrupt for ADCF0 | √ | √ | √ | √ |
| EI | EIINT81 | ADCF | Trigger group for SG0 of ADCF0 | √ | √ | √ | √ |
| EI | EIINT82 | ADCF | Trigger group for SG1 of ADCF0 | √ | √ | √ | √ |
| EI | EIINT83 | ADCF | Trigger group for SG2 of ADCF0 | √ | √ | √ | √ |
| EI | EIINT84 | ADCF | Trigger group for SG3 of ADCF0 | √ | √ | √ | √ |
| EI | EIINT85 | ADCF | Trigger group for SG4 of ADCF0 | √ | √ | √ | √ |
| EI | EIINT86 | CSIH | Transmission Interrupt Reception Error signal for CSIH_0 | √ | √ | √ | √ |
| EI | EIINT87 | CSIH | Transmission Interrupt Reception signal for CSIH_0 | √ | √ | √ | √ |
| EI | EIINT88 | CSIH | Transmission Interrupt Communication signal for CSIH_0 | √ | √ | √ | √ |
| EI | EIINT89 | CSIH | Transmission Interrupt for JOB Completion for CSIH_0 | √ | √ | √ | √ |
| EI | EIINT90 | CSIH | Transmission Interrupt Reception Error signal for CSIH_1 | √ | √ | √ | √ |
| EI | EIINT91 | CSIH | Transmission Interrupt Reception signal for CS0 of CSIH_1 | √ | √ | √ | √ |
| EI | EIINT92 | CSIH | Transmission Interrupt Communication signal for CSIH_1 | √ | √ | √ | √ |
| EI | EIINT93 | CSIH | Transmission Interrupt for JOB Completion for CSIH_1 | √ | √ | √ | √ |
| EI | EIINT94 | CSIH | Transmission Interrupt Reception Error signal for CSIH_2 | √ | √ | √ | √ |
| EI | EIINT95 | CSIH | Transmission Interrupt Reception signal for CSIH_2 | √ | √ | √ | √ |
| EI | EIINT96 | CSIH | Transmission Interrupt Communication signal for CSIH_2 | √ | √ | √ | √ |
| EI | EIINT97 | CSIH | Transmission Interrupt for JOB Completion for CSIH_2 | √ | √ | √ | √ |
| EI | EIINT98 | CSIH | Transmission Interrupt Reception Error signal for CSIH_3 | √ | √ | √ | √ |
| EI | EIINT99 | CSIH | Transmission Interrupt Reception signal for CSIH_3 | √ | √ | √ | √ |
| EI | EIINT100 | CSIH | Transmission Interrupt Communication signal for CSIH_3 | √ | √ | √ | √ |

Table 6.26 Difference among P1M-C, P1H-C and P1H-CE (4/7)

| Level | Symbol | Source | Generating Condition or Factor | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|---|
| EI | EIINT101 | CSIH | Transmission Interrupt for JOB Completion for CSIH_3 | √ | √ | √ | √ |
| EI | EIINT102 | RLIN3 | RLIN3_0 interrupt 0 status | √ | √ | √ | √ |
| EI | EIINT103 | RLIN3 | RLIN3_0 interrupt 0 receive | √ | √ | √ | √ |
| EI | EIINT104 | RLIN3 | RLIN3_0 interrupt 0 transfer | √ | √ | √ | √ |
| EI | EIINT105 | RLIN3 | RLIN3_1 interrupt 0 status | √ | √ | √ | √ |
| EI | EIINT106 | RLIN3 | RLIN3_1 interrupt 0 receive | √ | √ | √ | √ |
| EI | EIINT107 | RLIN3 | RLIN3_1 interrupt 0 transfer | √ | √ | √ | √ |
| EI | EIINT108 | RLIN3 | RLIN3_2 interrupt 0 status | — | √*3 | √ | √ |
| EI | EIINT109 | RLIN3 | RLIN3_2 interrupt 0 receive | — | √*3 | √ | √ |
| EI | EIINT110 | RLIN3 | RLIN3_2 interrupt 0 transfer | — | √*3 | √ | √ |
| EI | EIINT111 | RLIN3 | RLIN3_3 interrupt 0 status | — | √*3 | √ | √ |
| EI | EIINT112 | RLIN3 | RLIN3_3 interrupt 0 receive | — | √*3 | √ | √ |
| EI | EIINT113 | RLIN3 | RLIN3_3 interrupt 0 transfer | — | √*3 | √ | √ |
| EI | EIINT114 | | | N/A | N/A | N/A | N/A |
| EI | EIINT115 | | | N/A | N/A | N/A | N/A |
| EI | EIINT116 | HS-USART | Bundled error interrupt for HS-USRT0 | √ | √ | √ | √ |
| EI | EIINT117 | HS-USART | Bundled reception interrupt for HS-USRT0 | √ | √ | √ | √ |
| EI | EIINT118 | HS-USART | Bundled transmission interrupt for HS-USRT0 | √ | √ | √ | √ |
| EI | EIINT119 | HS-USART | Bundled error interrupt for HS-USRT1 | √ | √ | √ | √ |
| EI | EIINT120 | HS-USART | Bundled reception interrupt for HS-USRT1 | √ | √ | √ | √ |
| EI | EIINT121 | HS-USART | Bundled transmission interrupt for HS-USRT1 | √ | √ | √ | √ |
| EI | EIINT122 | HS-USART | Bundled error interrupt for HS-USRT2 | — | √*3 | √ | √ |
| EI | EIINT123 | HS-USART | Bundled reception interrupt for HS-USRT2 | — | √*3 | √ | √ |
| EI | EIINT124 | HS-USART | Bundled transmission interrupt for HS-USRT2 | — | √*3 | √ | √ |
| EI | EIINT125 | HS-USART | Bundled error interrupt for HS-USRT3 | — | √*3 | √ | √ |
| EI | EIINT126 | HS-USART | Bundled reception interrupt for HS-USRT3 | — | √*3 | √ | √ |
| EI | EIINT127 | HS-USART | Bundled transmission interrupt for HS-USRT3 | — | √*3 | √ | √ |
| EI | EIINT128 | Pin | External Interrupt5 | √ | √ | √ | √ |
| EI | EIINT129 | Pin | External Interrupt6 | √ | √ | √ | √ |
| EI | EIINT130 | Pin | External Interrupt7 | √ | √ | √ | √ |
| EI | EIINT131 | Pin | External Interrupt8 | — | √*3 | √ | √ |
| EI | EIINT132 | Pin | External Interrupt9 | √*1*3 | √*3 | √ | √ |
| EI | EIINT133 | Pin | External Interrupt10 | √ | √ | √ | √ |
| EI | EIINT134 | Pin | External Interrupt11 | — | √*3 | √ | √ |
| EI | EIINT135 | STM | STM interrupt 6 (STM0) | √ | √ | √ | √ |
| EI | EIINT136 | STM | STM interrupt 7 (STM0) | √ | √ | √ | √ |
| EI | EIINT137 | STM | STM interrupt 4 (STM1) | — | √ | √ | √ |
| EI | EIINT138 | STM | STM interrupt 5 (STM1) | — | √ | √ | √ |
| EI | EIINT139 | STM | STM interrupt 6 (STM1) | — | √ | √ | √ |
| EI | EIINT140 | STM | STM interrupt 7 (STM1) | — | √ | √ | √ |
| EI | EIINT141 | GTM | TIM Shared interrupts (TIM0_IRQ1) | √ | √ | √ | √ |
| EI | EIINT142 | GTM | TIM Shared interrupts (TIM0_IRQ3) | √ | √ | √ | √ |
| EI | EIINT143 | GTM | TIM Shared interrupts (TIM0_IRQ5) | √ | √ | √ | √ |

Table 6.26 Difference among P1M-C, P1H-C and P1H-CE (5/7)

| Level | Symbol | Source | Generating Condition or Factor | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|---|
| EI | EIINT144 | GTM | TIM Shared interrupts (TIM0_IRQ7) | √ | √ | √ | √ |
| EI | EIINT145 | GTM | TIM Shared interrupts (TIM1_IRQ1) | √ | √ | √ | √ |
| EI | EIINT146 | GTM | TIM Shared interrupts (TIM1_IRQ3) | √ | √ | √ | √ |
| EI | EIINT147 | GTM | TIM Shared interrupts (TIM1_IRQ5) | √ | √ | √ | √ |
| EI | EIINT148 | GTM | TIM Shared interrupts (TIM1_IRQ7) | √ | √ | √ | √ |
| EI | EIINT149 | GTM | MCS Interrupt for channel (MCS0_IRQ1) | √ | √ | √ | √ |
| EI | EIINT150 | GTM | MCS Interrupt for channel (MCS0_IRQ3) | √ | √ | √ | √ |
| EI | EIINT151 | GTM | MCS Interrupt for channel (MCS0_IRQ5) | √ | √ | √ | √ |
| EI | EIINT152 | GTM | MCS Interrupt for channel (MCS0_IRQ7) | √ | √ | √ | √ |
| EI | EIINT153 | GTM | MCS Interrupt for channel (MCS1_IRQ0) | — | √ | √ | √ |
| EI | EIINT154 | GTM | MCS Interrupt for channel (MCS1_IRQ2) | — | √ | √ | √ |
| EI | EIINT155 | GTM | MCS Interrupt for channel (MCS1_IRQ4) | — | √ | √ | √ |
| EI | EIINT156 | GTM | ATOM Shared interrupts (ATOM0_IRQ1) | √ | √ | √ | √ |
| EI | EIINT157 | GTM | ATOM Shared interrupts (ATOM0_IRQ3) | √ | √ | √ | √ |
| EI | EIINT158 | GTM | ATOM Shared interrupts (ATOM0_IRQ5) | √ | √ | √ | √ |
| EI | EIINT159 | GTM | ATOM Shared interrupts (ATOM0_IRQ7) | √ | √ | √ | √ |
| EI | EIINT160 | GTM | ATOM Shared interrupts (ATOM1_IRQ1) | √ | √ | √ | √ |
| EI | EIINT161 | GTM | ATOM Shared interrupts (ATOM1_IRQ3) | √ | √ | √ | √ |
| EI | EIINT162 | GTM | ATOM Shared interrupts (ATOM1_IRQ5) | — | √ | √ | √ |
| EI | EIINT163 | GTM | ATOM Shared interrupts (ATOM1_IRQ7) | — | √ | √ | √ |
| EI | EIINT164 | GTM | ATOM Shared interrupts (ATOM2_IRQ1) | — | √ | √ | √ |
| EI | EIINT165 | GTM | ATOM Shared interrupts (ATOM2_IRQ3) | — | √ | √ | √ |
| EI | EIINT166 | ADCF | Error Interrupt for ADCF1 | √ | √ | √ | √ |
| EI | EIINT167 | ADCF | Trigger group for SG0 of ADCF1 | √ | √ | √ | √ |
| EI | EIINT168 | ADCF | Trigger group for SG1 of ADCF1 | √ | √ | √ | √ |
| EI | EIINT169 | ADCF | Trigger group for SG2 of ADCF1 | √ | √ | √ | √ |
| EI | EIINT170 | ADCF | Trigger group for SG3 of ADCF1 | √ | √ | √ | √ |
| EI | EIINT171 | ADCF | Trigger group for SG4 of ADCF1 | √ | √ | √ | √ |
| EI | EIINT172 | MCAN | interrupt0 for MTTCAN0 | √ | √ | √ | √ |
| EI | EIINT173 | MCAN | interrupt1 for MTTCAN0 | √ | √ | √ | √ |
| EI | EIINT174 | MCAN | filter event for MTTCAN0 | √ | √ | √ | √ |
| EI | EIINT175 | MCAN | interrupt0 for MCAN0 | √ | √ | √ | √ |
| EI | EIINT176 | MCAN | interrupt1 for MCAN0 | √ | √ | √ | √ |
| EI | EIINT177 | MCAN | filter event for MCAN0 | √ | √ | √ | √ |
| EI | EIINT178 | MCAN | interrupt0 for MCAN1 | √ | √ | √ | √ |
| EI | EIINT179 | MCAN | interrupt1 for MCAN1 | √ | √ | √ | √ |
| EI | EIINT180 | MCAN | Filter event for MCAN1 | √ | √ | √ | √ |
| EI | EIINT181 | MCAN | interrupt0 for MCAN2 | — | — | √ | √ |
| EI | EIINT182 | MCAN | interrupt1 for MCAN2 | — | — | √ | √ |
| EI | EIINT183 | MCAN | filter event for MCAN2 | — | — | √ | √ |
| EI | EIINT184 | Ethernet | Ethernet interrupt for ETH0 | √ | √ | √ | √ |
| EI | EIINT185 | Ethernet | Ethernet interrupt for ETH1 | — | √*3 | √ | √ |
| EI | EIINT186 | FlexRay | Universal interrupt chA for FLX0 | √ | √ | √ | √ |

Table 6.26 Difference among P1M-C, P1H-C and P1H-CE (6/7)

| Level | Symbol | Source | Generating Condition or Factor | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|---|
| EI | EIINT187 | FlexRay | Universal interrupt chB for FLX0 | √ | √ | √ | √ |
| EI | EIINT188 | FlexRay | Timer 0 interrupt for FLX0 | √ | √ | √ | √ |
| EI | EIINT189 | FlexRay | Timer 1 interrupt for FLX0 | √ | √ | √ | √ |
| EI | EIINT190 | FlexRay | Timer 2 interrupt for FLX0 | √ | √ | √ | √ |
| EI | EIINT191 | FlexRay | FIFO data available (FIFO not empty) interrupt for FLX0 | √ | √ | √ | √ |
| EI | EIINT192 | FlexRay | FIFO warning interrupt for FLX0 | √ | √ | √ | √ |
| EI | EIINT193 | FlexRay | Output transfer warning interrupt for FLX0 | √ | √ | √ | √ |
| EI | EIINT194 | FlexRay | Output transfer done interrupt for FLX0 | √ | √ | √ | √ |
| EI | EIINT195 | FlexRay | Input queue full interrupt for FLX0 | √ | √ | √ | √ |
| EI | EIINT196 | FlexRay | Input queue empty interrupt for FLX0 | √ | √ | √ | √ |
| EI | EIINT197 | FlexRay | Universal interrupt chA for FLX1 | — | √*3 | √ | √ |
| EI | EIINT198 | FlexRay | Universal interrupt chB for FLX1 | — | √*3 | √ | √ |
| EI | EIINT199 | FlexRay | Timer 0 interrupt for FLX1 | — | √*3 | √ | √ |
| EI | EIINT200 | FlexRay | Timer 1 interrupt for FLX1 | — | √*3 | √ | √ |
| EI | EIINT201 | FlexRay | Timer 2 interrupt for FLX1 | — | √*3 | √ | √ |
| EI | EIINT202 | FlexRay | FIFO data available (FIFO not empty) interrupt for FLX1 | — | √*3 | √ | √ |
| EI | EIINT203 | FlexRay | FIFO warning interrupt for FLX1 | — | √*3 | √ | √ |
| EI | EIINT204 | FlexRay | Output transfer warning interrupt for FLX1 | — | √*3 | √ | √ |
| EI | EIINT205 | FlexRay | Output transfer done interrupt for FLX1 | — | √*3 | √ | √ |
| EI | EIINT206 | FlexRay | Input queue full interrupt for FLX1 | — | √*3 | √ | √ |
| EI | EIINT207 | FlexRay | Input queue empty interrupt for FLX1 | — | √*3 | √ | √ |
| EI | EIINT208 | SENT | status interrupt for SENT_0 | √ | √ | √ | √ |
| EI | EIINT209 | SENT | receive interrupt for SENT_0 | √ | √ | √ | √ |
| EI | EIINT210 | SENT | status interrupt for SENT_1 | √ | √ | √ | √ |
| EI | EIINT211 | SENT | receive interrupt for SENT_1 | √ | √ | √ | √ |
| EI | EIINT212 | SENT | status interrupt for SENT_2 | √ | √ | √ | √ |
| EI | EIINT213 | SENT | receive interrupt for SENT_2 | √ | √ | √ | √ |
| EI | EIINT214 | SENT | status interrupt for SENT_3 | √ | √ | √ | √ |
| EI | EIINT215 | SENT | receive interrupt for SENT_3 | √ | √ | √ | √ |
| EI | EIINT216 | SENT | status interrupt for SENT_4 | √ | √ | √ | √ |
| EI | EIINT217 | SENT | receive interrupt for SENT_4 | √*3 | √*3 | √ | √ |
| EI | EIINT218 | SENT | status interrupt for SENT_5 | √*3 | √*3 | √ | √ |
| EI | EIINT219 | SENT | receive interrupt for SENT_5 | √*3 | √*3 | √ | √ |
| EI | EIINT220 | SENT | status interrupt for SENT_6 | — | √*3 | √ | √ |
| EI | EIINT221 | SENT | receive interrupt for SENT_6 | — | √*3 | √ | √ |
| EI | EIINT222 | SENT | status interrupt for SENT_7 | — | √*3 | √ | √ |
| EI | EIINT223 | SENT | receive interrupt for SENT_7 | — | √*3 | √ | √ |
| EI | EIINT224 | SENT | status interrupt for SENT_8 | — | — | — | √ |
| EI | EIINT225 | SENT | receive interrupt for SENT_8 | — | — | — | √ |
| EI | EIINT226 | SENT | status interrupt for SENT_9 | — | — | — | √ |
| EI | EIINT227 | SENT | receive interrupt for SENT_9 | — | — | — | √ |

Table 6.26 Difference among P1M-C, P1H-C and P1H-CE (7/7)

| Level | Symbol | Source | Generating Condition or Factor | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|---|
| EI | EIINT228 | DTS | DTS transmission complete interrupt for ch0 to 31 | √ | √ | √ | √ |
| EI | EIINT229 | DTS | DTS transmission complete interrupt for ch32 to 63 | √ | √ | √ | √ |
| EI | EIINT230 | DTS | DTS transmission complete interrupt for ch64 to 95 | √ | √ | √ | √ |
| EI | EIINT231 | DTS | DTS transmission complete interrupt for ch96 to 127 | √ | √ | √ | √ |
| EI | EIINT232 | DTS | DTS count match interrupt for ch0 to 31 | √ | √ | √ | √ |
| EI | EIINT233 | DTS | DTS count match interrupt for c32 to 63 | √ | √ | √ | √ |
| EI | EIINT234 | DTS | DTS count match interrupt for ch64 to 95 | √ | √ | √ | √ |
| EI | EIINT235 | DTS | DTS count match interrupt for ch96 to 127 | √ | √ | √ | √ |
| EI | EIINT236 | BHP | interrupt for degradation | √ | √ | √ | √ |
| EI | EIINT237 | FICUIF | FENTRYC bit update interrupt for FICUIF_0 | √ | √ | √ | √ |
| EI | EIINT238 | FICUIF | FENTRYD bit update interrupt for FICUIF_0 | √ | √ | √ | √ |
| EI | EIINT239 | FICUIF | FENTRYC bit update interrupt for FICUIF_1 | √ | √ | √ | √ |
| EI | EIINT240 | Pin | Edge detect of RLIN3 0 received data | √ | √ | √ | √ |
| EI | EIINT241 | Pin | Edge detect of RLIN3 1 received data | √ | √ | √ | √ |
| EI | EIINT242 | Pin | Edge detect of MTTCAN0 received data | √ | √ | √ | √ |
| EI | EIINT243 | Pin | Edge detect of MCAN0 received data | √ | √ | √ | √ |
| EI | EIINT244 | Pin | Edge detect of FlexRay 0 ch A received data | √ | √ | √ | √ |
| EI | EIINT245 | Pin | Edge detect of FlexRay 1 ch A received data | — | √*3 | √ | √ |
| EI | EIINT246 | | | N/A | N/A | N/A | N/A |
| EI | EIINT247 | | | N/A | N/A | N/A | N/A |
| EI | EIINT248 | | | N/A | N/A | N/A | N/A |
| EI | EIINT249 | | | N/A | N/A | N/A | N/A |
| EI | EIINT250 | | | N/A | N/A | N/A | N/A |
| EI | EIINT251 | | | N/A | N/A | N/A | N/A |
| EI | EIINT252 | | | N/A | N/A | N/A | N/A |
| EI | EIINT253 | | | N/A | N/A | N/A | N/A |
| EI | EIINT254 | | | N/A | N/A | N/A | N/A |
| EI | EIINT255 | | | N/A | N/A | N/A | N/A |

Note 1. Not in P1M-C (QFP-144)

Note 2. The registers for FEINT.
FEINTF0, FEINTFC0,FEINTFMSK0 are available for all.
FEINTF1, FEINTFC1,FEINTFMSK1 are not available for P1M-C

Note 3. Not in P1M-C (BGA-156) / P1H-C (4MB, BGA-156)

# Section 7 DMA

## 7.1 Overview

### 7.1.1 Overview

Direct memory access (DMA) is used to access data without going through the CPU.

DMA consists of two types of DMA transfer modules: DMAC and DTS. A DMAC includes registers for storing transfer information, and a DTS stores transfer information in the dedicated RAM (DTSRAM). DMA has two 8-channel DMAC modules and one 128-channel DTS module.

In this manual, DTFR denotes the function to select among hardware DMA transfer sources for a DMAC and retain the DMA transfer request, and DTSFSL denotes the function to retain a DMA transfer request for each DTS channel. The DTFR can handle 128 types of hardware DMA transfer sources, and the DTSFSL can handle 128 types of them.

The address space that can be used for DMA transfer is a 4 GB address space represented by a 32-bit address. For information about which resource is assigned to a particular area in the 4 GB address space and which area is accessible from DMA, see **Section 4, Address Map**.

### 7.1.2 Clock Supply

Clock supply by and to the DMA is listed in the following table.

**Table 7.1 Clock Supply**

| Unit Name | Clock for the Unit | Supply Clock Name |
|---|---|---|
| DMA/DTS | Operation clock (DMA clock) | CLK_HSB |

### 7.1.3 Term Definition

**Table 7.2** shows the terms used in this section.

Table 7.2 List of Term Definitions

| Term | Meaning |
|---|---|
| DMA transfer | A general term for data transfer carried out by DMA. |
| DMA cycle | A series of actions that consist of reading an amount of data specified by the transfer size (8/16/32/64/128 bits) from the address specified by the source address and writing it to the address specified by the destination address. The first half of the DMA cycle (reading part) is called a read cycle, and the second half (writing part) is called a write cycle. |
| Hardware DMA transfer source | A trigger for a DMA transfer request given by an internal peripheral device. |
| Hardware DMA transfer request | A DMA transfer request generated by a hardware DMA transfer source |
| Software DMA transfer request | A DMA transfer request generated by software writing to a register. |
| DMA transfer request | A trigger to start DMA transfer with a DMAC or DTS. |
| Transfer information (TI) | The information required for DMA transfer, including the source address, destination address, transfer data size, and transfer count. The transfer information for a DTS is specifically termed as TI. |
| DTSRAM | RAM used by a DTS to store the transfer information. |
| Single transfer | DMA transfer consisting of one DMA cycle started by one DMA transfer request. |
| Block transfer 1 | DMA transfer consisting of a number of DMA cycles specified by the transfer count in the transfer information, started by one DMA transfer request. |
| Block transfer 2 | DMA transfer consisting of a number of DMA cycles specified by the address reload count in the transfer information, started by one DMA transfer request. |
| Block transfer | A general term for both block transfer 1 and block transfer 2. |
| Last transfer | The DMA cycle carried out when the transfer count in the transfer information is 1. |
| Address reload transfer | The DMA cycle carried out when the address reload count in the transfer information is 1 if the reload function 2 is used. |
| Suspension | An action of pausing DMA transfer during block transfer. You can resume DMA transfer after suspension. |
| Resume | An action of resuming suspended DMA transfer. |
| Transfer abort | An action of aborting DMA transfer in the middle. You cannot resume DMA transfer after that. |

# 7.2 DMA Function

## 7.2.1 Basic Operation of DMA Transfer

### 7.2.1.1 Transfer Mode

DMA has three transfer modes.

#### Single Transfer

One DMA cycle is executed when a DMA transfer request is accepted.

#### Block Transfer 1

A number of DMA cycles specified in the transfer count register are executed when a DMA transfer request is accepted.

#### Block Transfer 2

A number of DMA cycles specified by the address reload count are executed when a DMA transfer request is accepted. If the address reload count is larger than the value in the transfer count register, a number of DMA cycles specified in the transfer count register are executed.

### 7.2.1.2 Executing a DMA Cycle

DMA always executes a write cycle after a read cycle is complete. For example, if the transfer data size is 128 bits, a write cycle is executed after a read cycle for the 128-bit data is finished. A write cycle never starts in the middle of a read cycle.
The bus is not locked. Consequently, a CPU cycle may interrupt between the read and write cycles, and between the four read cycles and four write cycles during multi-byte transfers. In other words, the DMA cycles are not atomic. Source and destination addresses must be aligned according to the transfer width

### 7.2.1.3 Updating Transfer Information

When a DMA cycle is executed, DMA updates transfer information as follows.

#### Source Address and Destination Address

Transfer information will be updated as described in **Table 7.3** according to the settings for the source address and destination address and the settings in the transfer control register such as the count directions of source address and destination address and transfer data size.

**Table 7.3 Updating the Source Address and the Destination Address (1/2)**

| Direction of Count | Transfer Data Size | Address after Update |
|---|---|---|
| Increment | 8 bit | (address before update) + $0000_0001_H$ |
| | 16 bit | (address before update) + $0000_0002_H$ |
| | 32 bit | (address before update) + $0000_0004_H$ |
| | 64 bit | (address before update) + $0000_0008_H$ |
| | 128 bit | (address before update) + $0000_0010_H$ |

Table 7.3 Updating the Source Address and the Destination Address (2/2)

| Direction of Count | Transfer Data Size | Address after Update |
|---|---|---|
| Decrement | 8 bit | (address before update) – $0000_0001_H$ |
| | 16 bit | (address before update) – $0000_0002_H$ |
| | 32 bit | (address before update) – $0000_0004_H$ |
| | 64 bit | (address before update) – $0000_0008_H$ |
| | 128 bit | (address before update) – $0000_0010_H$ |
| Fixed | — | Same as the address before update. |

When you use the reload function, a special update rule is applied other than the one described in **Table 7.3** for the last transfer and the address reload transfer. For details, see **Section 7.2.3, Reload Function**.

#### Transfer Count/Address Reload Count

The transfer count is decremented by one for every DMA cycle.

The address reload count is decremented by one for every DMA cycle when the reload function 2 or block transfer 2 is used. When the reload function 2 or block transfer 2 is not used, it is not updated.

When you use the reload function, a special update rule is applied for the last transfer and the address reload transfer. For details, see **Section 7.2.3, Reload Function**.

#### Other transfer information

Other transfer information is not updated during execution of a DMA cycle.

### 7.2.1.4 Last Transfer and Address Reload Transfer

The last transfer means a DMA cycle executed when the value in the transfer count register, which shows the remaining number of transfers, is one. The last transfer differs in operation compared to other DMA cycles as follows.

- The transfer completion flag (DCSTn.TC) is set when the last transfer is complete. (Only applicable for a DMAC)
- The channel operation enable (DCENn.DTE) bit is cleared when the last transfer is complete. (Only applicable for a DMAC. When the continuous transfer is disabled.)
- When the transfer completion interrupt output enable is set, a transfer completion interrupt is output when the last transfer is complete.
- When the reload function 1 is enabled, the reload function 1 is executed at the timing of the last transfer. For details, see **Section 7.2.3, Reload Function**.

The address reload transfer means a DMA cycle executed when the reload function 2 is enabled and the address reload count is one. The reload function 2 is executed during the address reload transfer. For details, see **Section 7.2.3, Reload Function**.

### 7.2.1.5 Transfer Completion Interrupt and Transfer Count Match Interrupt Outputs

DMA can output a transfer completion interrupt and a transfer count match interrupt.

#### Transfer Completion Interrupt Output

When the transfer completion interrupt output enable (DTCTn.TCE) is set in the transfer control register, a DMAC requests a DMAC transfer completion interrupt when the last transfer is complete.

When the transfer completion interrupt output enable (DTTCTn.TCE) is set in the transfer control register, a DTS requests a DTS transfer completion interrupt when the last transfer is complete.

#### Transfer Count Match Interrupt Output

When the transfer count match interrupt enable (DTCTn.CCE) is set in the transfer control register, a DMAC requests a DMAC transfer count match interrupt when the DMA cycle in which the transfer count compare register and the transfer count have the same value is complete.

When the transfer count match interrupt enable (DTTCTn.CCE) is set in the transfer control register, a DTS requests a DTS transfer count match interrupt at the completion of the DMA cycle in which the transfer count compare register and the transfer count have the same value.

**Figure 7.1** shows the operation of the transfer completion interrupt and the transfer count match interrupt.

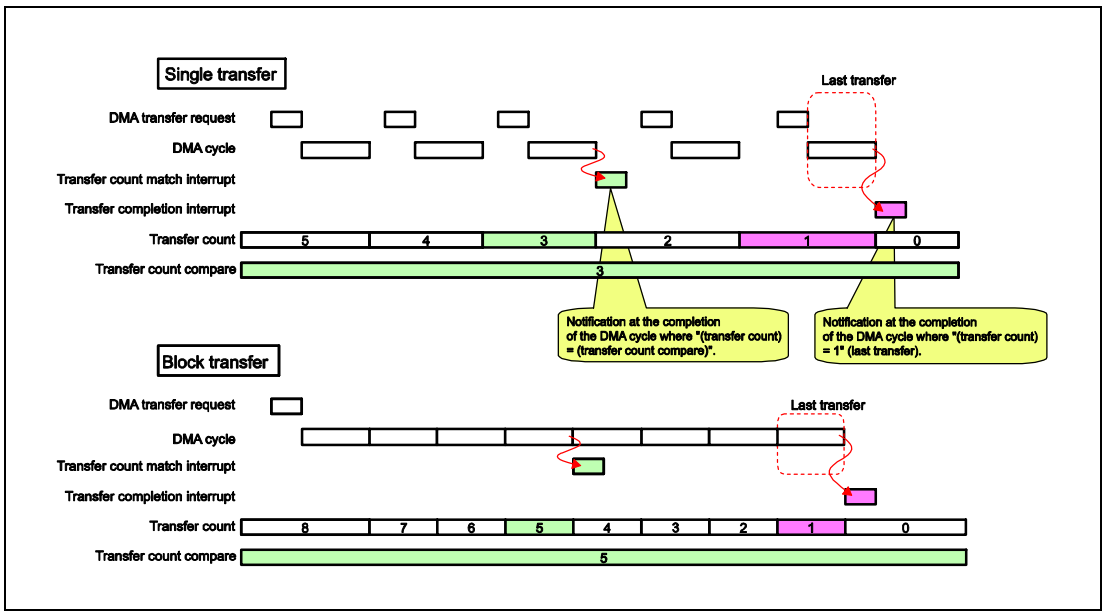



Figure 7.1 Transfer Completion Interrupt and Transfer Count Match Interrupt

#### 7.2.1.6 Continuous Transfer

If the continuous transfer is not used, a DMAC sets the transfer completion flag (DCSTn.TC) and clears the channel operation enable (DCENn.DTE) bit when the last transfer is complete. In this case, a DMA transfer request is not accepted when the request is generated after the completion of the last transfer.

If the continuous transfer is used, the channel operation enable (DCENn.DTE) bit is not cleared when the last transfer is complete, and a DMA transfer request can be accepted even when the transfer completion flag is set. If DMA is used for a case where a specified number of DMA transfers are executed repetitively, software overhead associated with clearing the transfer completion flag and setting the channel operation enable bit after the completion of the last transfer can be reduced by using the continuous transfer.

The continuous transfer is enabled by setting the continuous transfer enable (DTCTn.MLE) in the DMAC transfer control register.

The continuous transfer is designed to work with the reload function 1. The continuous transfer function itself does not update the source address register, destination address register, and transfer count register. If, after the last transfer is complete, you want to restore the source address register, destination address register, and transfer count register to the state before the DMA transfer starts, use the reload function 1 and set the values of the source address register, destination address register, and the transfer count register before the DMA transfer starts to the reload source address register, reload destination address register, and reload transfer count register respectively.

A DTS does not have a setting corresponding to the continuous transfer enable (DTCTn.MLE) for a DMAC. This is because a DTS does not have bits like the transfer completion flag (DCSTn.TC) and the channel operation enable (DCENn.DTE) a DMAC has.

A DTS does not start DMA transfer when a DMA transfer request is generated while the transfer count is 0. (This corresponds to the case for a DMAC where the continuous transfer is not used.)

If the reload function 1 is used for a DTS and the value other than 0 is reloaded to the transfer count when the last transfer is complete, DMA transfer can start when the next DMA transfer request is accepted. (This corresponds to the case for a DMAC where the continuous transfer is used.)

**Figure 7.2** shows an operation of continuous transfer by a DMAC.

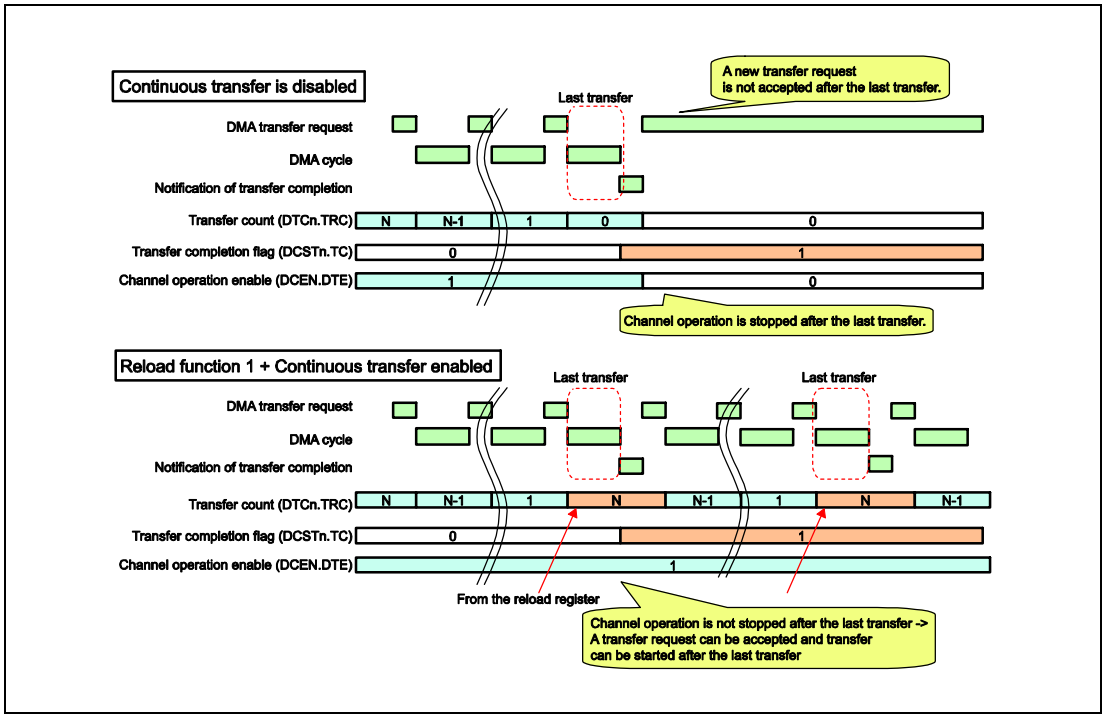



**Figure 7.2 Operation of Continuous Transfer by a DMAC**

## 7.2.2 Channel Priority Order

This subsection explains arbitration between multiple DMA channels.

### 7.2.2.1 DMAC Channel Arbitration

A DMAC select one channel out of eight channels with arbitration. Arbitration is done according to the fixed priority order. The priority order is “channel 0 > channel 1 > channel 2 > channel 3 > channel 4 > channel 5 > channel 6 > channel 7” for DMAC0, and “channel 8 > channel 9 > channel 10 > channel 11 > channel 12 > channel 13 > channel 14 > channel 15” for DMAC1.

Arbitration is done for every DMA cycle. No arbitration occurs between the read and write of a DMA cycle.

If, at the timing when one DMA cycle in the middle of a block transfer of a channel is complete, there is a DMA transfer request from a channel with a higher priority than the channel, a DMA cycle of the channel with the higher priority will be executed next as the result of arbitration.

If a DMAC executes the block transfer 1 or block transfer 2, DMAC channel arbitration is done for every DMA cycle, and possibly a DMA cycle of another DMAC channel with a higher priority can cut in.

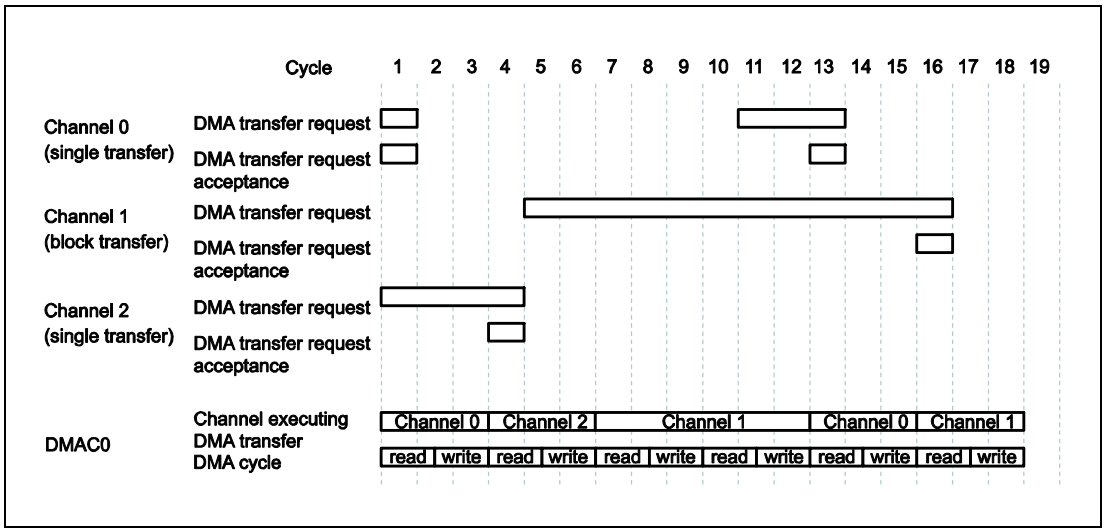



**Figure 7.3 DMAC Channel Arbitration**

Cycle numbers shown in **Figure 7.3** are for explanation purpose only. They do not indicate an actual number of cycles necessary for executing DMA transfer.

In **Figure 7.3**, DMA transfer requests for channels 0 and 2 are generated at Cycle 1. As a result of arbitration, a DMA cycle for channel 0 starts because its priority is higher. At Cycle 4, a DMA cycle for channel 2 starts. At Cycle 5, a DMA transfer request for channel 1 is generated. The DMA cycle for channel 2 is still ongoing and no arbitration is done at this point. At Cycle 7, a DMA cycle for channel 1 starts. Channel 1 uses block transfer. Another DMA cycle continues at Cycle 10 because there are no DMA transfer requests from other channels. At Cycle 11, a DMA transfer request for channel 0 is generated. The DMA cycle for channel 1 is still ongoing and no arbitration is done at this point. At Cycle 12, the DMA cycle for channel 1 is complete. At Cycle 13, a DMA cycle for channel 0 starts as a result of arbitration between DMA channels 0 and 1.

It should be noted that, even though a block transfer of channel 1 has been already started, a DMA cycle of not channel 1 but channel 0 is executed at Cycle 13 because the priority of the latter is higher.

At Cycle 15, the DMA cycle for channel 0 is complete. At Cycle 16, a DMA cycle for channel 1 starts again. At Cycle 18, the last DMA cycle of the block transfer of channel 1 is complete.

#### 7.2.2.2 DTS Channel Arbitration

If there are DMA transfer requests from multiple DTS channels, the DTSFSL arbitrates those DTS channels. For each DTS channel, a priority can be selected from four levels using DTS channel priority setting registers.

If there are DMA transfer request from multiple DTS channels, the arbitration is done as follows.

1. A channel with a higher priority level in the setting of DTS channel priority setting registers has a priority.
2. If two channels have the same priority level in the setting of DTS channel priority setting registers, a channel with a lower channel number has a priority.

The DTSFSL sends the DTS a DMA transfer request for the channel selected by arbitration. The DTS executes DMA transfer when it accepts the DMA transfer request.

Unlike DMA transfer with a DMAC, DMA transfer with a DTS does not allow arbitration between DTS channels in the middle of a block transfer. That means, even if a DMA transfer request with a higher priority comes during a block transfer for a channel with a lower priority, the DMA transfer with a higher priority does not start until the current block transfer for the channel with a lower priority is complete*.

**Note 1.** Block transfer is complete when the last transfer for the block transfer 1 or the last transfer or address reload transfer for the block transfer 2 occurs.

When a DTS executes the block transfer 1 or block transfer 2, a DMA cycle of a DTS channel with a higher priority does not take over the ongoing block transfer until the last transfer.

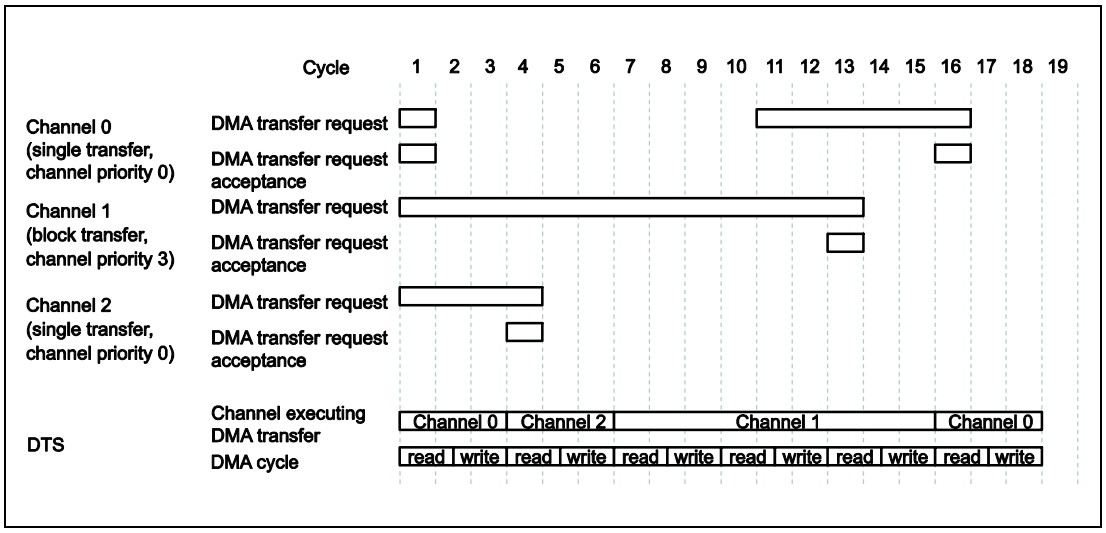



**Figure 7.4 DTS Channel Arbitration**

Cycle numbers shown in **Figure 7.4** are for explanation purpose only. They do not indicate an actual number of cycles necessary for executing DMA transfer.

In **Figure 7.4**, DMA transfer requests for channels 0, 1, and 2 are generated at Cycle 1. The channel priority for channels 0 and 2 is 0 and is higher than the channel priority for channel 1, which is 3. In addition, if two channels have the same priority, the channel with the smaller channel number has a higher priority. Consequently, the priority order for arbitration is “channel 0 > channel 2 > channel 1”,

and a DMA cycle for channel 0 starts because its priority is the highest At Cycle 4, as a result of arbitration between channels 1 and 2, a DMA cycle for channel 2 starts. At Cycle 7, a DMA cycle for channel 1 starts. Channel 1 uses block transfer. Another DMA cycle continues at Cycle 10 because there are no DMA transfer requests from other channels. At Cycle 11, a DMA transfer request for channel 0 is generated. The DMA cycle for channel 1 is still ongoing and no arbitration is done until the block transfer of channel 1 is complete.

At Cycle 15, the block transfer of channel 1 is complete. At Cycle 16, a DMA cycle for channel 0 starts.

#### 7.2.2.3 Interface Arbitration

The interface between the DMASS and the System Interconnect is shown in **Figure 7.5**. DMAC0, DMAC1 and DTS operate independently and they can request bus transfers via the DMAT to the system interconnect. As the system interconnect performs non-blocking data transfers, multiple DMA transfers can be active coincidentally.

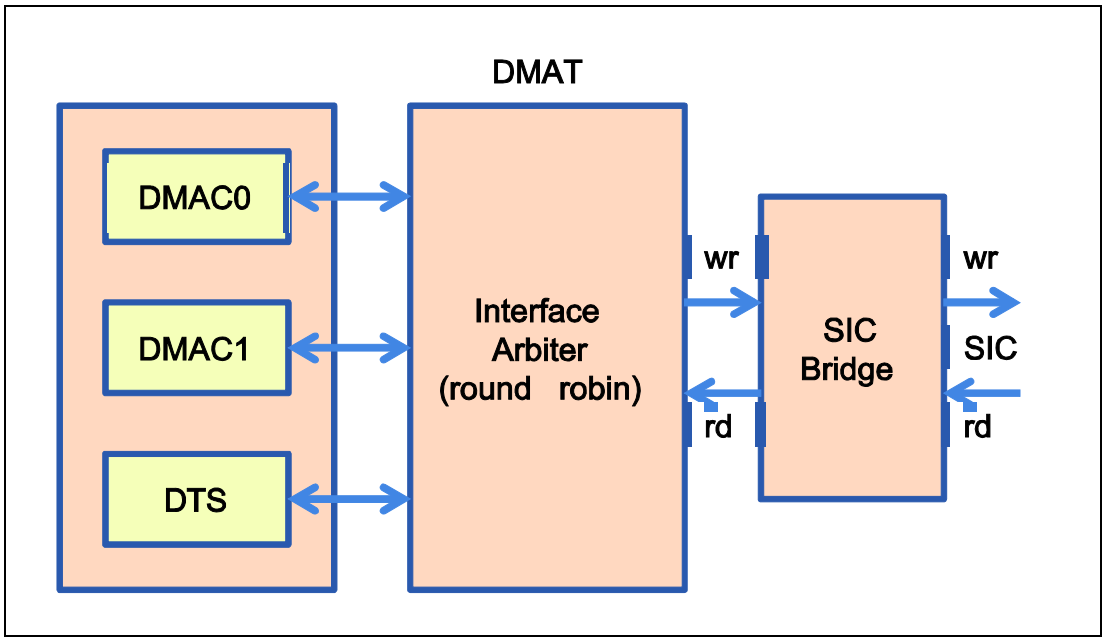



**Figure 7.5 Interface Arbitration**

Read and write busses are separated and they operate independently. In case of conflicts, the arbitration is done in round robin. Issuing a read or write request takes one DMA clock cycle. While the issuer waits for the response, another active unit may issue its request in the subsequent clock cycle.

## 7.2.3 Reload Function

### 7.2.3.1 Overview of the Reload Function

The reload function updates a portion of transfer information, more specifically, the source address, destination address, transfer count, and address reload count, to the predefined values during DMA transfer.

The reload function has two types of functions: reload function 1 and reload function 2.

### 7.2.3.2 Operation of Reload Function 1

When the reload function 1 is enabled, actions described in **Table 7.4** are executed at the timing of the last transfer according to the reload function 1 setting.

**Table 7.4 Operation of Reload Function 1**

| Reload Function 1 Setting | Register | Action at the Last Transfer |
|---|---|---|
| 00<br>(Reload function 1 disabled.) | Source address | Not reloaded. |
| | Destination address | Not reloaded. |
| | Transfer count | Not reloaded. |
| | Address reload count | Not reloaded. |
| 01<br>(Reload function 1 enabled. Reloading source address and transfer count.) | Source address | The reload source address is copied to this. |
| | Destination address | Not reloaded. |
| | Transfer count | The reload transfer count is copied to this. |
| | Address reload count | • If the reload function 2 is disable: Not reloaded.<br>• If the reload function 2 is enabled: The reload address reload count is copied to this. |
| 10<br>(Reload function 1 enabled. Reloading destination address and transfer count.) | Source address | Not reloaded. |
| | Destination address | The reload destination address is copied to this. |
| | Transfer count | The reload transfer count is copied to this. |
| | Address reload count | • If the reload function 2 is disable: Not reloaded.<br>• If the reload function 2 is enabled: The reload address reload count is copied to this. |
| 11<br>(Reload function 1 enabled. Reloading source address, destination address, and transfer count.) | Source address | The reload source address is copied to this. |
| | Destination address | The reload destination address is copied to this. |
| | Transfer count | The reload transfer count is copied to this. |
| | Address reload count | • If the reload function 2 is disable: Not reloaded<br>• If the reload function 2 is enabled: The reload address reload count is copied to this. |

**Figure 7.6** shows an operation of the reload function 1.

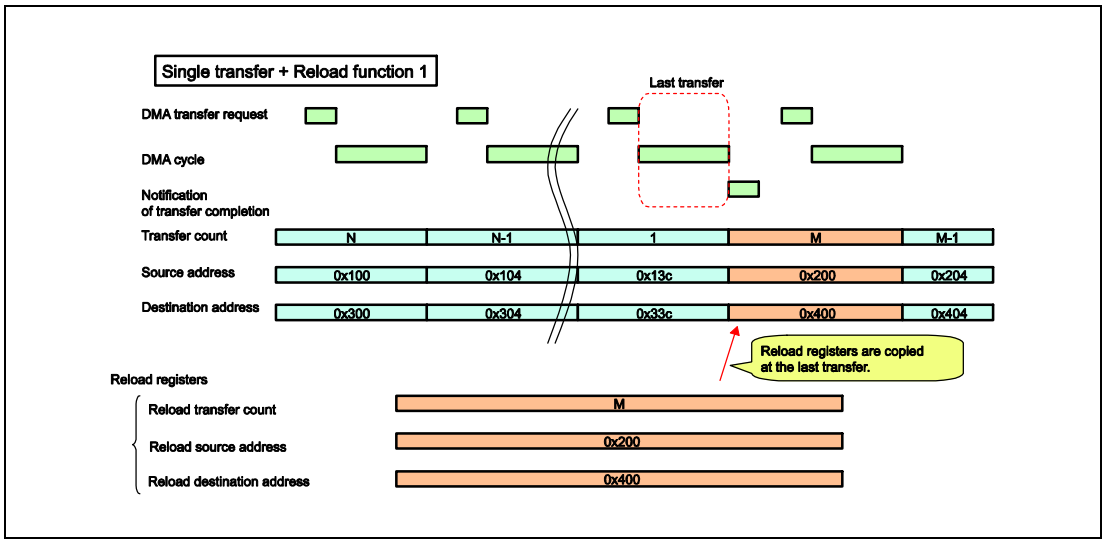



**Figure 7.6 Operation of Reload Function 1**

### 7.2.3.3 Reload Function 2

When the reload function 2 is enabled, actions described in **Table 7.5** are executed at the timing of the address reload transfer according to the reload function 2 setting.

**Table 7.5 Operation of Reload Function 2**

| Reload Function 2 Setting | Register | Action at the Address Reload Transfer |
|---|---|---|
| 00<br>(Reload function 2 disabled.) | Source address | Not reloaded. |
| | Destination address | Not reloaded. |
| | Address reload count | Not reloaded. |
| 01<br>(Reload function 2 enabled.<br>Reloading source address.) | Source address | The reload source address is copied to this. |
| | Destination address | Not reloaded. |
| | Address reload count | The reload address reload count is copied to this. |
| 10<br>(Reload function 2 enabled.<br>Reloading destination address.) | Source address | Not reloaded. |
| | Destination address | The reload destination address is copied to this. |
| | Address reload count | The reload address reload count is copied to this. |
| 11<br>(Reload function 2 enabled.<br>Reloading source address and destination address.) | Source address | The reload source address is copied to this. |
| | Destination address | The reload destination address is copied to this. |
| | Address reload count | The reload address reload count is copied to this. |

**Figure 7.7** shows an operation of the reload function 2.

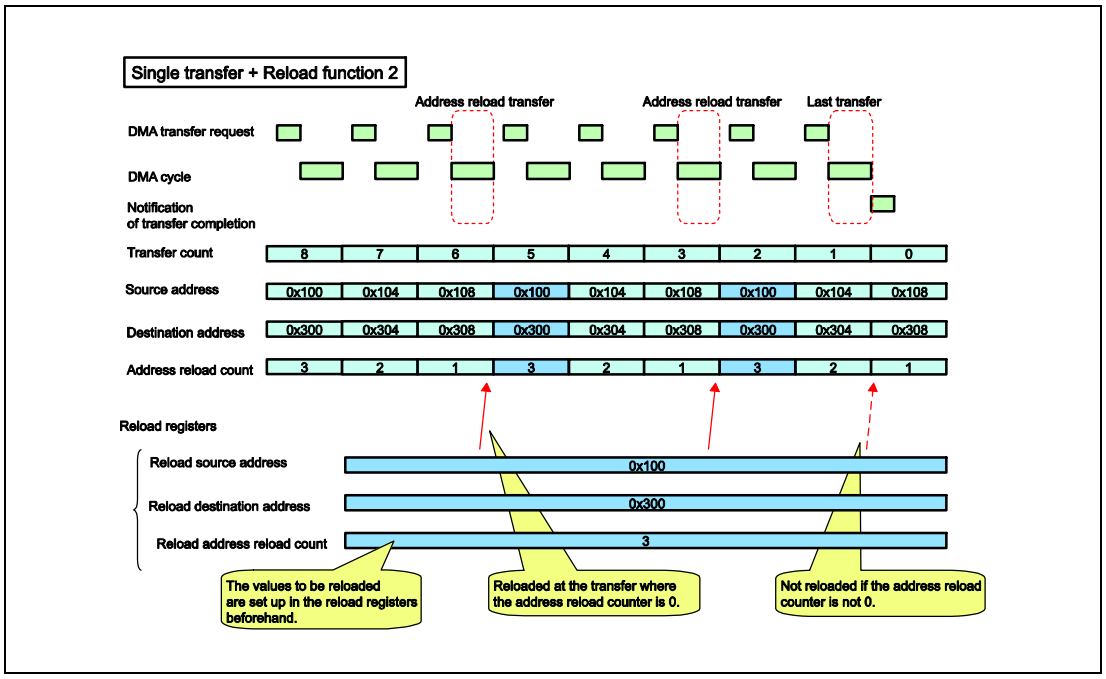



Figure 7.7 Operation of Reload Function 2

**Figure 7.8** shows an operation when both the reload function 1 and the reload function 2 are used simultaneously.

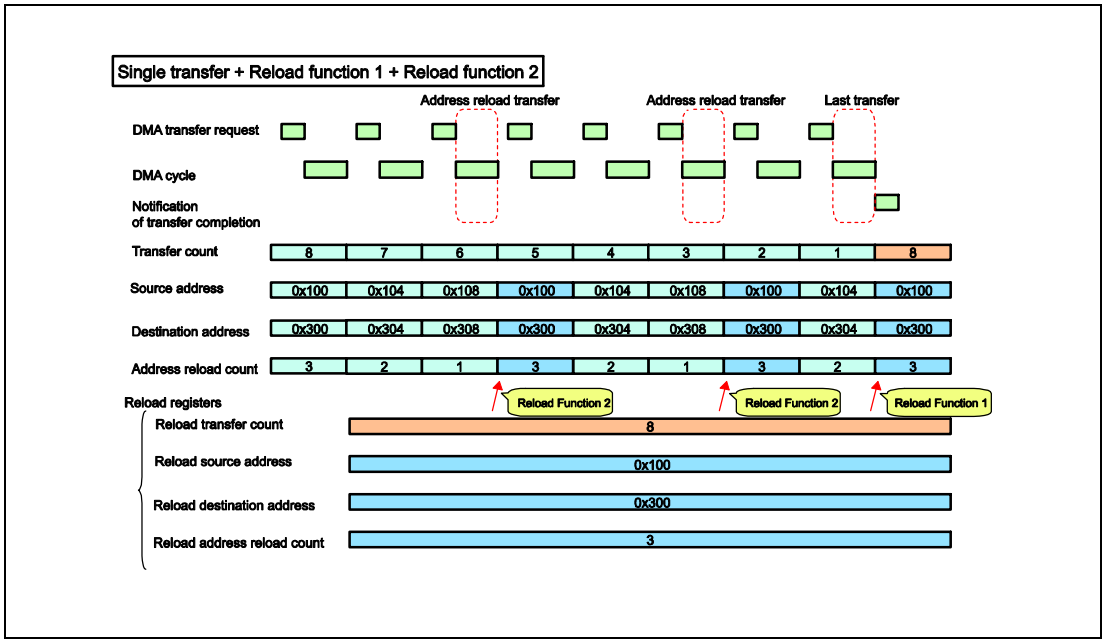



Figure 7.8 Operation when combining the reload function 1 and the reload function 2

#### 7.2.3.4 Timing of Setting DMAC Reload Registers

You can set up the reload source address register, reload destination address register, and reload transfer count register any time (even during DMA transfer). As an exception, if you update the reload source address register, reload destination address register, and reload transfer count register during DMA transfer, there may be a conflict between reloading at the last transfer or address reload transfer and updating the reload register. In order to avoid this conflict, setting up reload registers must be completed before the last transfer or address reload transfer starts.

If you need to update the reload source address register, reload destination address register, and reload transfer count register during DMA transfer, one way to know the right timing of update is to use a DMA transfer count match interrupt. In this case, you must set up the DMA transfer count compare register (DTCCn) so that you can have enough margin for the time necessary to update the reload registers.

#### 7.2.3.5 Timing of Setting DTS Reload Registers

It should be noted that the right timing of setting up the reload source address information, reload destination address information, and reload transfer count information differs depending on the transfer mode.

In the single transfer mode, the TI fetched at the beginning of the last transfer or address reload transfer is used for reload at the completion of the DMA cycle. Therefore, if you use the reload function for single transfer, the reload source address information, reload destination address information, and reload transfer count information in the TI must be set up before the beginning of the last transfer or address reload transfer.

During block transfer, TI is fetched only at the beginning of DMA transfer. The TI fetched at the beginning of the DMA transfer is used for reload at the last transfer or address reload transfer. Therefore, if you use the reload function for block transfer, the reload source address information, reload destination address information, and reload transfer count information in the TI must be set up before the beginning of the DMA transfer. If you update the reload source address information, reload destination address information, and reload transfer count information in the TI in the middle of a block transfer, those new settings will not be used for reload at the completion of the block transfer.

## 7.2.4 Chain Function

### 7.2.4.1 Overview

DMA offers a function called chain function. If you use the chain function, the completion of the DMA cycle or last transfer for one channel can trigger a DMA transfer request for another channel. A DMA transfer request for another channel initiated by the chain function is called a chain request.

You can select the condition for generating a chain request from the following two options.

- Always chain: A chain request is generated at the completion of every DMA cycle.
- Chain at the last transfer: A chain request is generated at the completion of the last transfer.

**Figure 7.9** shows an operation of the case “always chain”.

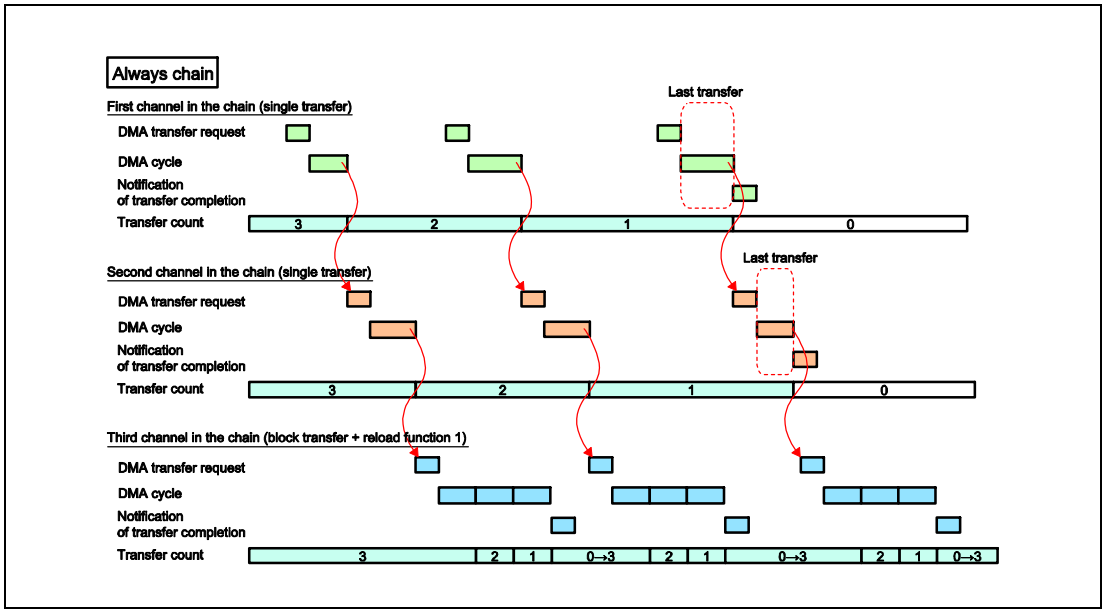



Figure 7.9 Operation of the Case “Always Chain”

shows an operation of the case “chain at the last transfer”.

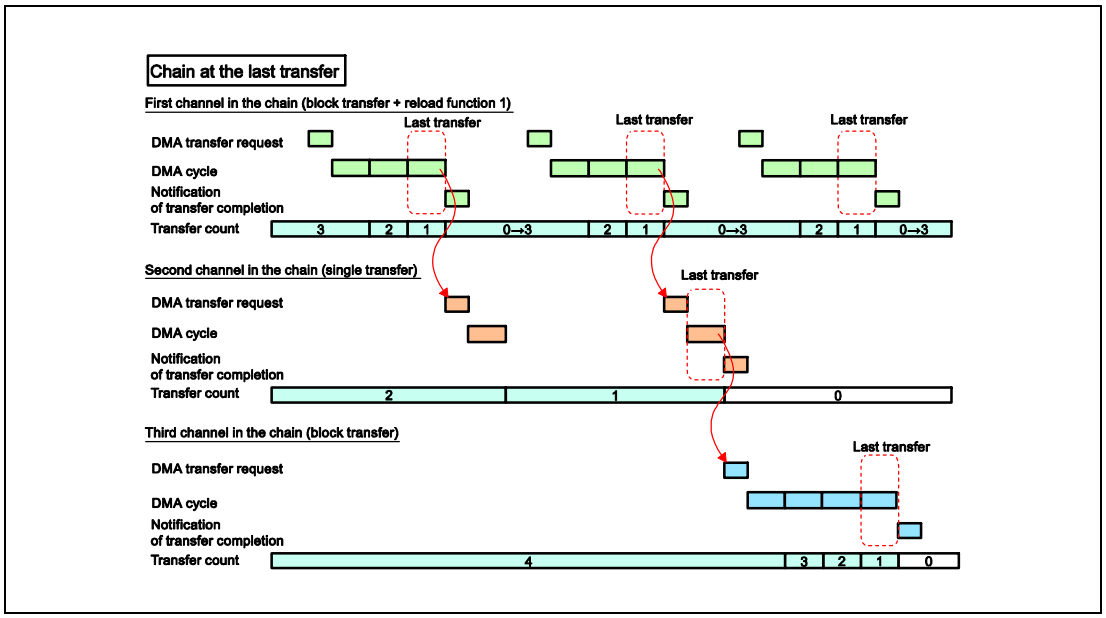



**Figure 7.10 Operation of the Case “Chain at the Last Transfer”.**

### 7.2.4.2 Setting Up the Chain Function

For a DMAC, you need to write to the chain enable (DTCTn.CHNE) and the next channel in the chain selection (DTCTn.CHNSEL) in the DMAC transfer control register in order to set up the type of chain function and the next channel number in the chain.

For a DTS, you need to write to the chain enable (DTTCTnnn.CHNE) and the next channel in the chain selection (DTTCTnnn.CHNSEL) in the DTS transfer control register in order to set up the type of chain function and the next channel number in the chain.

### 7.2.4.3 Caution for Using the Chain Function

The chain function sets the software DMA transfer request flag of the next channel in the chain as a part of its function. Therefore, you need to set up the channel settings of the next channel in the chain in the same way as when the software DMA transfer request is used. If you specify a channel using the hardware DMA transfer request for the next channel in the chain, the chain function does not work.

A channel and its next channel in the chain must belong to the same module (DMAC0, DMAC1, and DTS). You cannot specify a channel in another module for its next channel in the chain.

## 7.2.5 DMAC Operation

### 7.2.5.1 Types of DMA Transfer Requests and Assigning DMA Transfer Requests

A DMAC starts DMA transfer by accepting a hardware DMA transfer request or software DMA transfer request. The DMA transfer request selection assignment (DRS) bit in the DMAC transfer control register (DTCTn) determines whether a hardware DMA transfer request or a software DMA transfer request is used.

In the case of a hardware DMA transfer request for a DMAC, one out of 128 hardware DMA transfer sources is selected and assigned for each channel of the DMAC in the DTFR. This assignment is configured in the DTFR setting registers.

### 7.2.5.2 Generating and Accepting a Hardware DMA Transfer Request

DMAC can handle an edge-detection type of hardware DMA transfer source.

#### (1) Edge-detection type

In case of using edge-detection type, DTFR detects a rising edge of hardware DMA transfer source input and keeps it as a hardware DMA transfer request. DTFR also notifies DMAC of the existence of hardware DMA transfer request. When DTFR receives acknowledgement of hardware DMA transfer request from DMAC, DTFR clears hardware DMA transfer request.

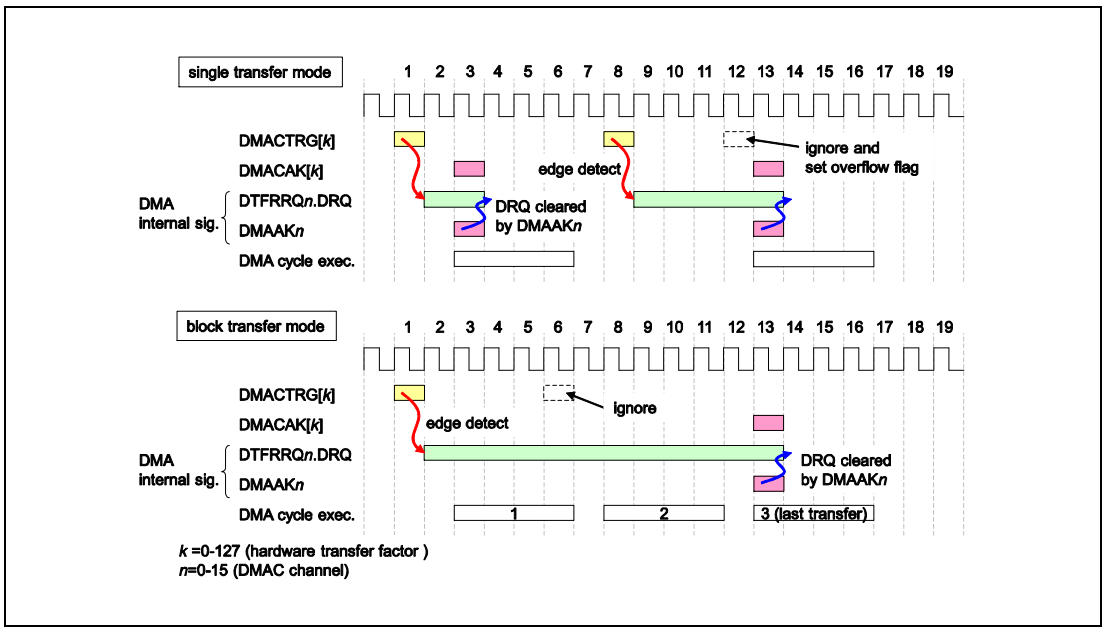



Figure 7.11 Operation of hardware DMA transfer request by DMAC (Edge-detection type)

In case of edge-detection type, DTFR can keep one hardware DMA transfer request per channel. If DTFR keeps a hardware DMA transfer request for a channel and additional rising edge of hardware DMA transfer source is inputted for the same channel, DTFR set the hardware DMA transfer request overflow flag of corresponding channel.

### (2) The Timing of Hardware DMA Transfer Acknowledgement Notification

The timing that DMA asserts the acknowledgement of hardware DMA transfer request (DMACAK) is different between DMA transfer modes.

- In case of single transfer, DMACAK is asserted at each time when DMA acknowledges hardware DMA transfer request.
- In case of block transfer 1, DMACAK is asserted when DMA executes final transfer.
- In case of block transfer 2, DMACAK is asserted when DMA executes address reload transfer.

**CAUTION**

**Caution about the operation of DTFR hardware DMA transfer source selection enable bit during block transfer.**

**Be careful about the following operation when DMAC is used both with hardware DMA transfer request and with block transfer (1 or 2) mode.**

**If DTFR hardware DMA transfer source selection enable bit is set to disable (DTFRn.REQEN = 0) by software while DMAC is executing block transfer, the ongoing block transfer is suspended.**

**When DMAC is used both with hardware DMA transfer request and with block transfer (1 or 2) mode, it is recommended not to modify DTFR hardware DMA transfer source selection enable bit (DTFRn.REQEN) by software while DMAC is executing block transfer. It is also recommended to use DTCT.MLE bit to control the acknowledgement of succeeding hardware DMA transfer requests.**

DTFR hardware DMA transfer request bit (DTFRRQn.DRQ) is kept set during block transfer and is cleared at the start of last transfer. (see the bottom figure of **Figure 7.12**). If DTFR hardware DMA transfer source selection enable bit is set to disable (DTFRn.REQEN = 0) by software while DMAC is executing block transfer, the hardware DMA transfer request is going to be masked and as a result DMAC cannot recognize the existence of hardware DMA transfer request.

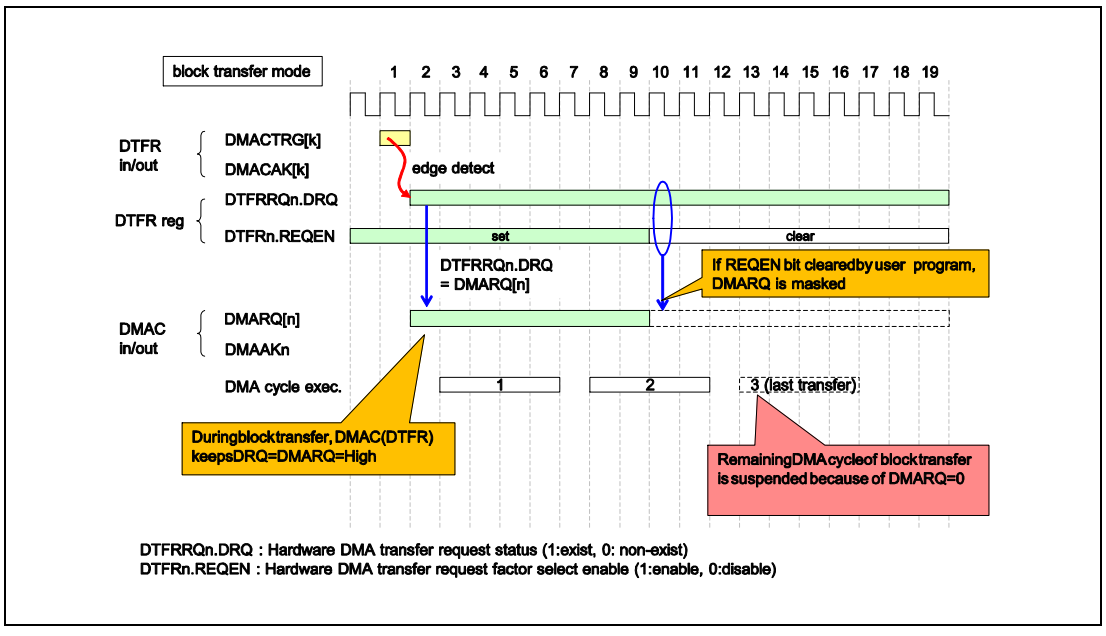



**Figure 7.12 Caution about the operation of DTFR hardware DMA transfer source selection enable bit during block transfer**

This caution does not apply to the following situations.

- The case that DMAC is used both with hardware DMA transfer request and with single transfer.
- The case that DMAC is used with software DMA transfer request.
- The case that DTS is used.

#### 7.2.5.3 Generating and Accepting a Software DMA Transfer Request

By setting the software DMA transfer request flag (SR) in the DMAC transfer status register (DCSTn) using the DMAC transfer status set register (DCSTSn), a software DMA transfer request can be generated.

The software DMA transfer request flag is automatically cleared when the DMAC processes the DMA transfer request. The timing when the software DMA transfer request flag is automatically cleared differs depending on the transfer mode of the DMA transfer to be executed.

- In the single transfer mode, the software DMA transfer request flag is cleared whenever the software DMA transfer request is accepted.
- In the block transfer 1 mode, the software DMA transfer request flag is cleared when the last transfer starts.
- In the block transfer 2 mode, the software DMA transfer request flag is cleared when the last transfer or address reload transfer starts.

The software DMA transfer request flag can also be cleared by software using the DMAC transfer status clear register (DCSTCn). When you abort a DMA transfer of a DMAC channel, you must clear the software DMA transfer request flag.

### 7.2.5.4 Execution of DMA transfer

The DMAC block executes the DMA transfer of the accepted channel. DMAC arbitration is done, when two or more channels request transfers at the same time.

**Figure 7.13** shows one example of the DMAC transfer timing in single transfer mode, starting with a transfer count of 2.

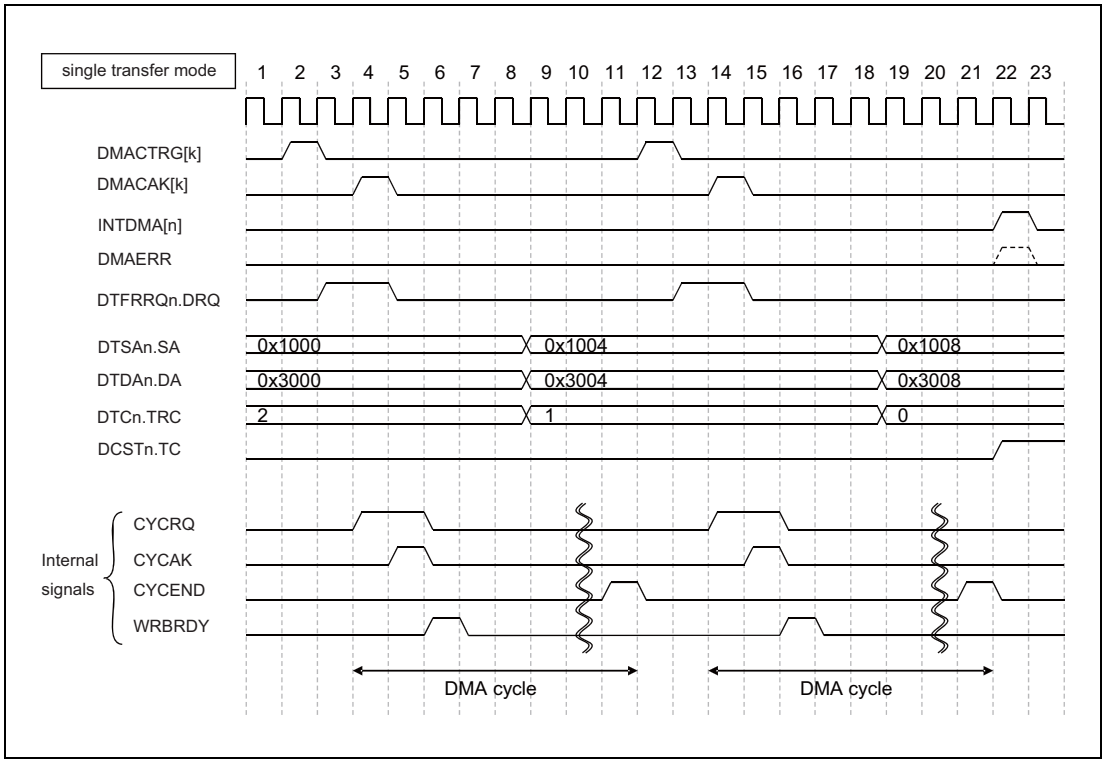



**Figure 7.13 Example of DMA transfer timing in single transfer mode**

**Figure 7.14** shows an example of the DMAC transfer timing in block transfer mode, starting with a transfer count of 2.

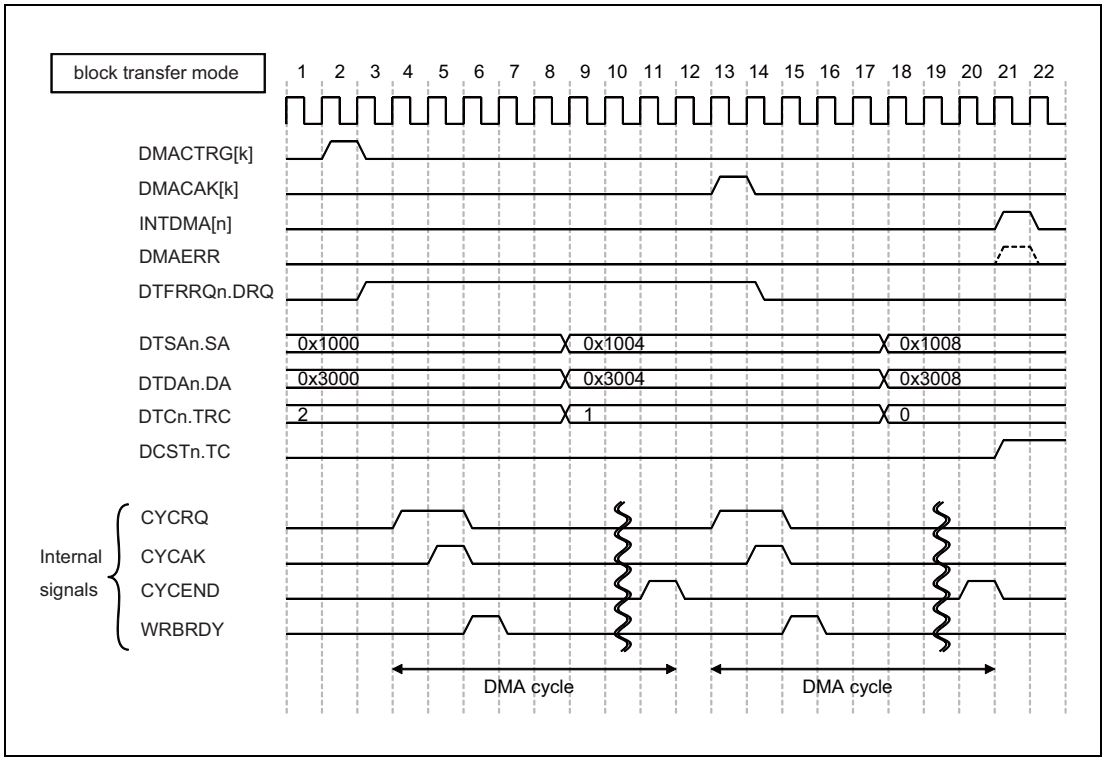



**Figure 7.14 Example of DMA transfer timing in block transfer mode**

**Figure 7.15** shows the timing, in which the source address register, the destination address register, the transfer count register and the transfer completion flag are updated.

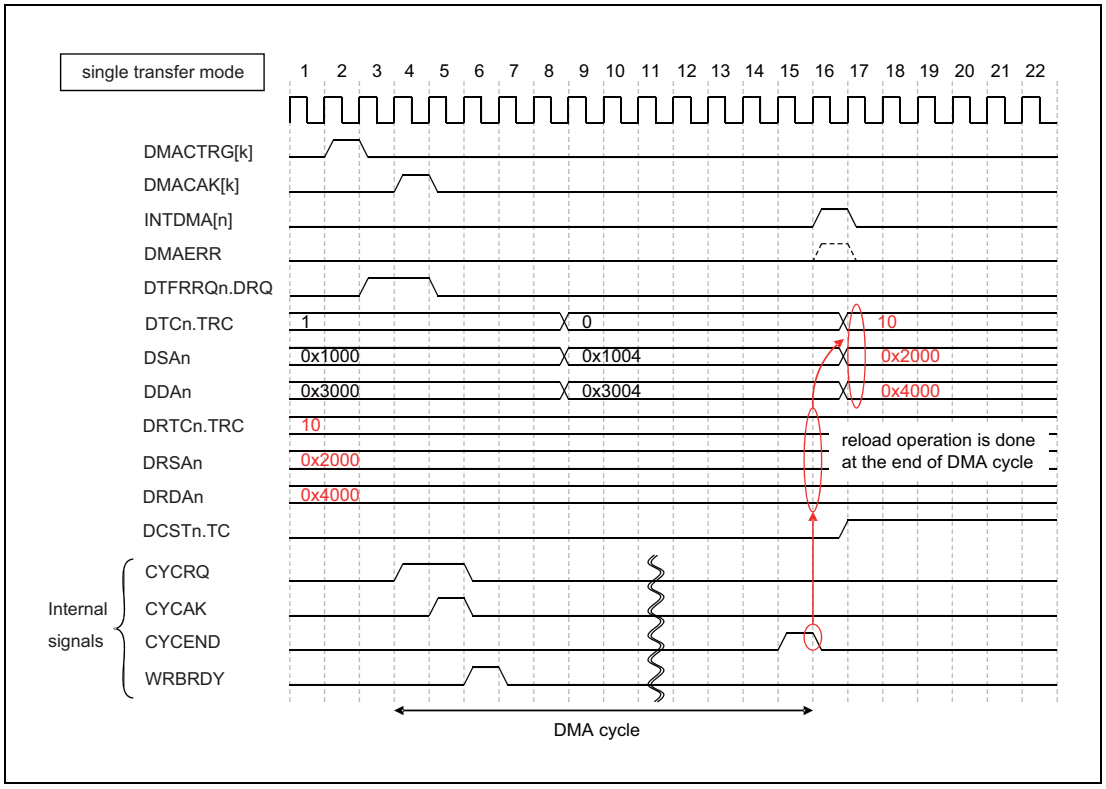



**Figure 7.15 Example of DMA transfer timing with reload function 1**

## 7.2.6 DTS Operation

### 7.2.6.1 Types of DMA Transfer Requests and Assigning DMA Transfer Requests

A DTS starts DMA transfer by accepting a hardware DMA transfer request or software DMA transfer request.

A transfer request for a DTS is retained in the transfer request pending bit of the DTSFSL for each channel.

As for the DTSFSL, both a hardware DMA transfer request and a software DMA transfer request are retained in the same transfer request pending bit. When executing DMA transfer, a DTS does not care whether a DMA transfer request is a hardware DMA transfer request or software DMA transfer request.

In the case of a hardware DMA transfer request for a DTS, each of 128 hardware DMA transfer sources is assigned to one of 128 channels in the DTSFSL in the fixed manner. You cannot change this assignment by, for example, register settings.

### 7.2.6.2 Generating and Accepting a DMA Transfer Request

When the DTSFSL detects a hardware DMA transfer source input, the DTSFSL sets the transfer request pending bit and retains the hardware DMA transfer source as a DMA transfer request. If the transfer request pending bit is set and the transfer request enable bit (DTFSLnnn.REQEN) in the DTSFSL operation setting register is set, the DTSFSL notifies the DTS of the DMA transfer request.

Software can also generate a DMA transfer request by setting the transfer request pending bit (DTFSTnnn.DRQ) using the DTSFSL transfer request set register (DTFSSnnn).

The DTSFSL can retain only one DMA transfer request per channel. If, while the transfer request pending bit for a channel is set, a new hardware DMA transfer source input for the same channel comes, DTSFSL sets request overflow flag of corresponding channel.

When the DTS accepts a DMA transfer request, it notifies of the acceptance of the DMA transfer request.

The transfer request pending bit is automatically cleared when the DTS accepts the DMA transfer request. The DTSFSL clears the transfer request pending bit automatically when the DTS accepts the DMA transfer request regardless of the type of the DMA transfer to be executed by the DTS.

The transfer request pending bit can also be cleared using the DTSFSL transfer request clear register (DTFSCnnn). If the transfer request pending bit of a channel is cleared before the DTS accepts the DMA transfer request, DMA transfer of the channel is not executed.

### 7.2.6.3 Executing DMA Transfer

When the DTS accepts a DMA transfer request for a channel, the DTS executes DMA transfer of the channel. If there are DMA transfer requests from multiple channels, the DTSFSL arbitrates the DTS channels and picks up one channel for a DMA transfer request.

While the DTS is executing DMA transfer, the DTS transfer status (DTSSTS.DTSACT) bit in the DTS status register is set. In addition, the channel number of the currently ongoing DMA transfer is set in the DTS transfer channel (DTSSTS.DTSACH).

When the DMA transfer is complete or aborted because of DMA transfer error or writing to registers and no channel is currently executing DMA transfer, the DTS transfer status (DTSSTS.DTSACT) bit is cleared.

**Figure 7.16** shows an example of a single DTS transfer. Note that the drawing shows also internal states, which are not reflected in registers. The DTS transfer is accepted in cycle 5, which causes fetching the transfer information (TI) in cycles 6 to 14. The DMA cycle is executed in cycles 15 to 23 and the TI information is written back in cycles 24 to 28. The state changes to idle and the DTS waits for the next DMA request. This picture is an example and the clock cycles of each phase may differ from the depicted flow.

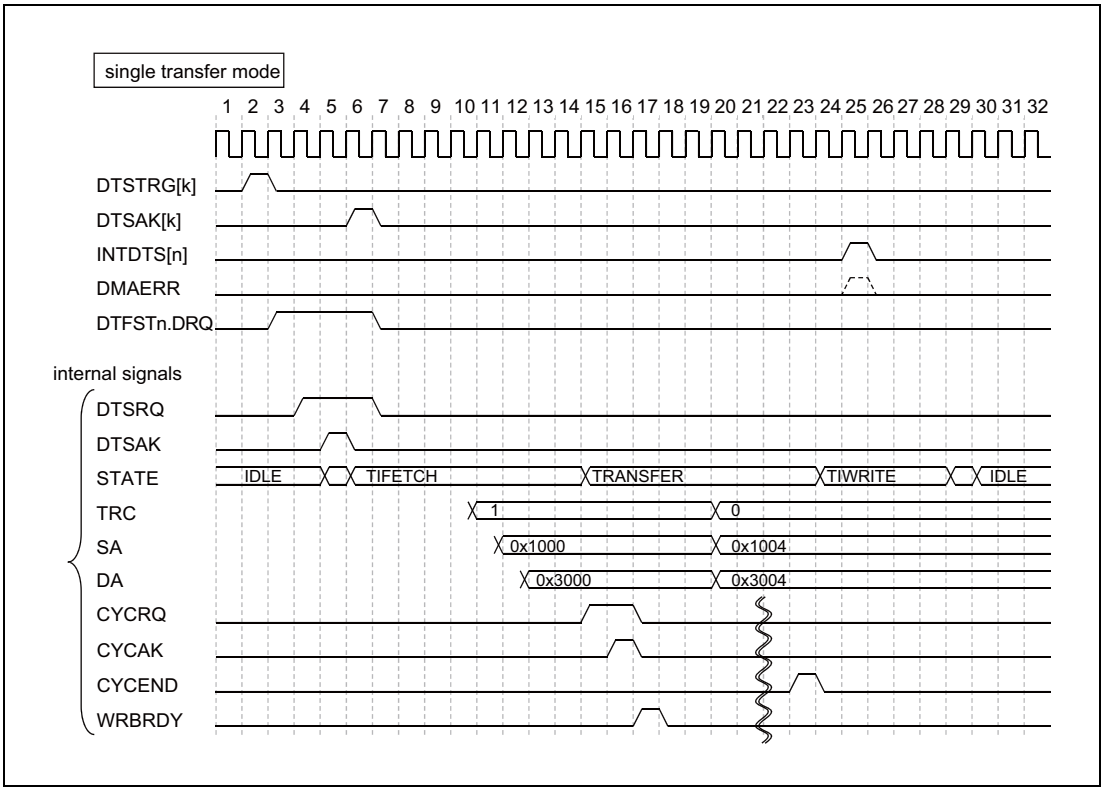



**Figure 7.16 DTS single transfer timing**

**Figure 7.17** shows an example of a DTS block transfer. The DTS transfer is accepted in cycle 5, which causes fetching the transfer information (TI) in cycles 6 to 10. Three transfers are executed in cycles 11 to 25 and the TI information is written back in cycles 26 to 30. In cycle 32, the state changes to idle and the next DMA transfer can be executed.

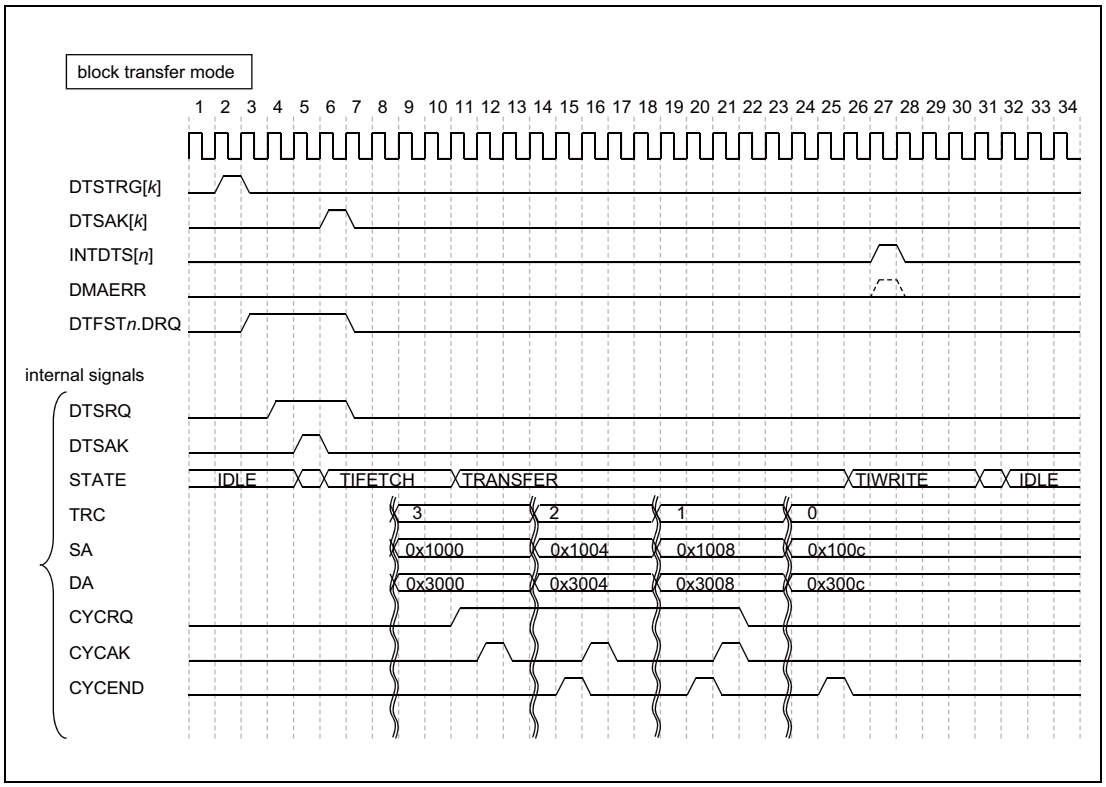



**Figure 7.17 DTS block transfer timing**

### 7.2.6.4 DTSRAM Access

A DTS accesses the DTSRAM when DMA transfer starts and finishes.

A DTS's action of reading transfer information from the DTSRAM when DMA transfer starts is called TI fetch.

A DTS's action of updating the transfer information on the DTSRAM when DMA transfer finishes is called TI write back.

A single transfer performs a TI fetch at the beginning of a DMA cycle and a TI write back at the end of a DMA cycle.

A block transfer performs a TI fetch at the beginning of the first DMA cycle and a TI write back at the end of the DMA cycle that satisfies the block transfer completion condition (the last transfer or address reload transfer).

Therefore, in the case of single transfer, the transfer information on the DTSRAM is updated for each DMA cycle. In the case of block transfer, the transfer information on the DTSRAM is updated after the completion of the block transfer. If software reads the transfer information on the DTSRAM during execution of a block transfer, the transfer information at the beginning of the block transfer is read.

**Table 7.6 Number of TI fetch cycles when not using the chain function**

| Transfer mode | Reload Function 1 | Reload Function 2 | Transfer count | Address reload count | TI fetch cycles |
|---|---|---|---|---|---|
| Single transfer | disabled | enabled | — | 1 | 12 cycles |
| | enabled | disabled | 1 | — | |
| | | enabled | >1 | 1 | |
| | | | 1 | — | |
| Block transfer 1 | disabled | enabled | — | Not zero | |
| | enabled | — | — | — | |
| Block transfer 2 | disabled | enabled | — | Not zero | |
| | enabled | — | — | — | |
| All other transfers | | | | | 9 cycles |

### 7.2.6.5 DTS Interrupt Merge function

A DTS interrupt merger maps 32 DTS interrupt request output signals into one interrupt request input signal of the interrupt controller. Four such mergers are implemented for the 128 DTS transfer completion interrupts and four others for the 128 DTS count match interrupts. The status of a DTS interrupt is available in the DTS interrupt status register (PINT0 to PINT7). Writing the DTS interrupt clear registers (PINTCLR0 to PINTCLR7) resets these flags.

**Table 7.7 DTS Interrupt-Related Registers**

| Interrupt Source | DTS Channel | DTS Interrupt Status Register | DTS Interrupt Clear Register |
|---|---|---|---|
| DTS transfer completion interrupt | 0 to 31 | PINT0 | PINTCLR0 |
| | 32 to 63 | PINT1 | PINTCLR1 |
| | 64 to 95 | PINT2 | PINTCLR2 |
| | 96 to 127 | PINT3 | PINTCLR3 |
| DTS transfer count match interrupt | 0 to 31 | PINT4 | PINTCLR4 |
| | 32 to 63 | PINT5 | PINTCLR5 |
| | 64 to 95 | PINT6 | PINTCLR6 |
| | 96 to 127 | PINT7 | PINTCLR7 |

#### (1) DTS Interrupt Processing Flow

**Figure 7.18** shows the DTS interrupt request flow.

- When only one interrupt request is generated out of bundled 32 interrupt sources
  - The bit corresponding to the interrupt request in the PINTn register is set to 1 and an interrupt request is output.
  - When the interrupt processing has been completed, write 1 to the clear register (PINTCLRn) to clear the interrupt request. The INTC waits for another interrupt request.
- When multiplex interrupt requests are generated out of bundled 32 interrupt sources
  - The highest-priority bit (interrupts on the lower-bits side take precedence) out of bits with interrupt request is extracted and only the extracted bit in the PINTn register is set to 1, and an interrupt request is output.
  - When the interrupt processing has been completed, write 1 to the clear register (PINTCLRn) to clear the highest-priority interrupt request.
  - After the highest-priority interrupt request has been cleared, the second highest-priority interrupt request is accepted. The corresponding bit in the PINTn register is set to 1 in the same way as before, and an interrupt request is output.
  - These steps are repeated until all interrupt sources bundled into 32 bits are cleared.

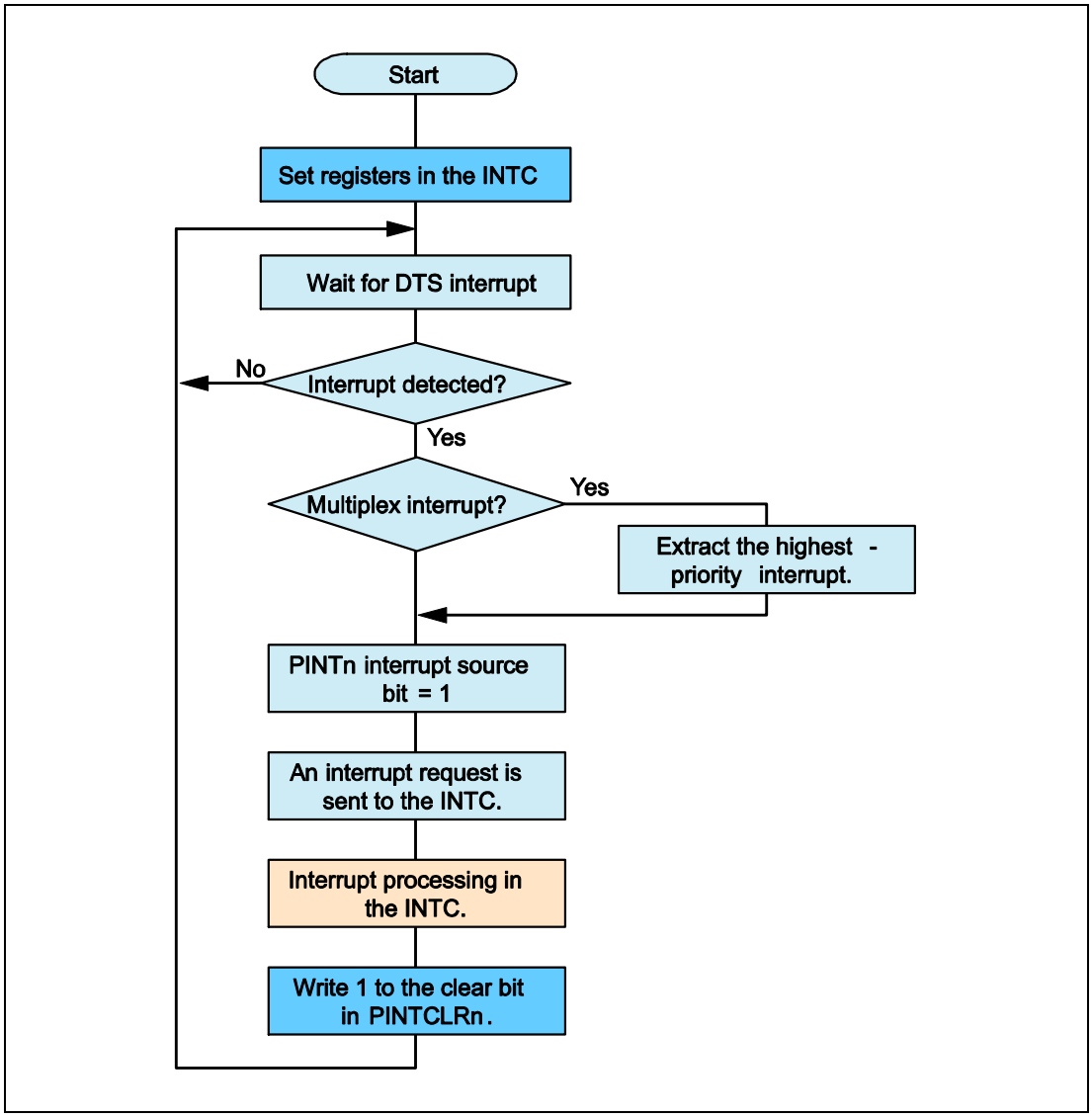



Figure 7.18 DTS Interrupt Processing Flow

#### 7.2.6.6 DTS Trigger Select function

Many DTS channels can be triggered by one of two alternative trigger sources. See **Table 7.13** for the supported assignments. The DTS select function selects the primary or secondary trigger input for each DTS channel.

**Table 7.8 DTS Trigger Select Registers**

| Register Name | Function | DTS Channels |
|---|---|---|
| DTSTRGSEL0 (See **Section 7.13.1.1**) | DTS primary/secondary select register 0 | 0 to 31 |
| DTSTRGSEL1 (See **Section 7.13.1.2**) | DTS primary/secondary select register 1 | 32 to 63 |
| DTSTRGSEL2 (See **Section 7.13.1.3**) | DTS primary/secondary select register 2 | 64 to 95 |
| DTSTRGSEL3 (See **Section 7.13.1.4**) | DTS primary/secondary select register 3 | 96 to 127 |

## 7.3 Temporarily Suspending DMA transfers

DMA transfers can be temporarily suspended by disabling individual channels or the whole DMA controller. All DMA transfers are suspended, by setting the DMACTL.DMASPD bit. Setting the DTSCTL1.DTSUST bit suspends DTS transfers. Each DMAC and DTS channel can hold one DMA request during executing DMA cycle, even if transfers are suspended or if the respective channel is disabled. Any additional transfer request is lost if a channel has already held a DMA request. Lost transfers due to such overflows are reported by the DTFRRQn.OVF and DTFSTnnn.OVF flags, again also during suspend and disable states.

### 7.3.1 Suspension, restart and abortion a DMAC channel

DMA transfers for individual channels can be suspended by clearing the DCENn.DTE bit of that channel. A DMA cycle, which is ongoing while DTE is cleared, will be finished. DMA transfers resume, when DTE is set. As one DMA transfer request is stored during suspension, it is recommended to clear the hardware DMA transfer request in the DTFR in the case of a hardware DMA transfer request, the software DMA request flag DCSTn.SR by writing the register flag clear command into DCSTCn.SRC in the case of a software DMA transfer request, if the DMA transfer shall be entirely stopped.

In case that the continuous transfer enable bit (DTCTn.MLE) is set, the channel operation enable bit (DCENn.DTE) is kept to be set. Even though the channel operation enable bit (DCENn.DTE) is cleared by software during a DMA cycle in a last transfer, the function of the continuous transfer enable bit (DTCTn.MLE) is given high priority and the channel operation enable bit (DCENn.DTE) is set after completion of the last transfer. If you want to abort an ongoing DMA transfer of a DMAC channel when continuous transfer function is enabled, please clear the continuous transfer enable bit (DTCTn.MLE) first and then clear the channel operation enable bit (DCENn.DTE) to abort DMA transfer of the DMAC channel. Only for the operation, DMAC Transfer Control Register (DTCTn) can be written under the channel operation is enabled (DCENn.DTE = 1).

**Figure 7.19** shows an example of suspending, restarting and aborting DMA transfers. Both channels 0 and 1 execute block transfers. Channel 1 begins its transfer at t1 and is interrupted by the higher priority channel 0 at t2. The channel 0 transfers end at t3 and channel 1 resumes its transfer, which ends at t4. At t5, channel 1 starts the next block transfer, which is again interrupted at t6 due to the higher priority of channel 0. At t7, channel 0 is suspended and therefore channel 1 can finish its block transfer at t8. At t9, channel 0 has been restarted and it resumes its block transfer until it is again suspended at t10. At t11, the DMA channel 0 is stopped and the ongoing block transfer is aborted. Therefore, no further transfer is started at t12.

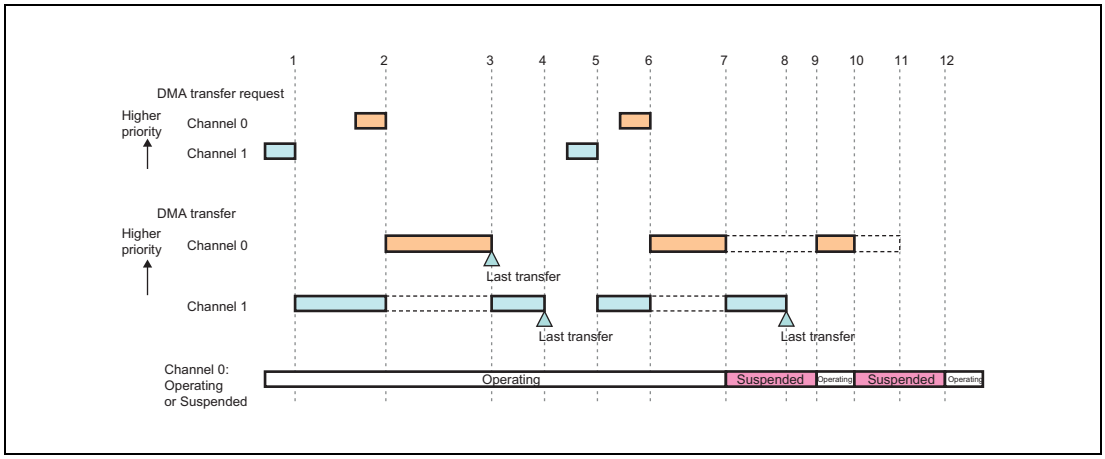



**Figure 7.19 Example of Suspension, Resume, and Transfer Abort of a DMAC Channel**

### 7.3.2 Suspension, Resume, and Transfer Abort of a DTS

You can suspend the DMA transfer executed by a DTS by setting the DTS suspend bit (DTSCTL1.DTSUST) in the DTS control register 1. If a DMA cycle is ongoing, the DMA transfer is suspended at the timing when the DMA cycle is finished. If the ongoing DMA cycle is a single transfer or a transfer that completes a block transfer (the last transfer or address reload transfer), the DMA transfer is suspended after a TI write back after the completion of the DMA cycle. If the ongoing DMA cycle is a type other than the above, the DMA transfer is suspended after the completion of the DMA cycle without a TI write back. If you resume the DMA transfer while the DMA transfer is suspended, clear the DTS suspend bit in the DTS control register 1.

If you want to abort the currently ongoing DMA transfer executed by a DTS, suspend the DTS as described above, and then set the DTS transfer abort request bit (DTSCTL2.DTSTIT) in the DTS control register 2 to abort the currently suspended DMA transfer. If transfer is aborted, no TI write back is executed. In addition, aborting the DMA transfer does not change the value of the DTS suspend bit (DTSCTL1.DTSUST). If you want the DTS to accept another DMA transfer request after the abort, clear the DTS suspend bit.

**Figure 7.20** shows an example of suspension, resume, and transfer abort of a DTS.

In **Figure 7.20**, channels 0, 1, and 2 are executing block transfer. At time tick 1, a DMA transfer request for channel 1 is accepted and DMA transfer starts. At time tick 2, DMA transfer requests for channels 0 and 2 are generated. At time tick 3, the last transfer of channel 1 is complete, and as a result of DTS channel arbitration, a DMA transfer request for channel 0 is accepted, and DMA transfer of channel 0 starts because channel 0 has a higher priority than channel 2. At time tick 4, the last transfer of channel 0 is complete, and DMA transfer of channel 2 starts. At time tick 5, the DTS is put into the suspended state, and the DMA transfer of channel 2 is suspended. At time tick 6, DMA transfer requests for channels 0 and 1 are generated. At time tick 7, the suspended state for the DTS is cleared, and the DMA transfer of channel 2, which has been suspended in the middle of a block transfer, is resumed. If DMA transfer is suspended in the middle of a block transfer, no DTS channel arbitration is done when it is resumed. At time tick 8, the last transfer of channel 2 is complete, and as a result of DTS channel arbitration, a DMA transfer request for channel 0 is accepted and the DMA transfer starts because channel 0 has a higher priority than channel 2. At time tick 9, the DTS is put into the suspended state, and at time tick 10, the suspended DMA transfer of channel is aborted. When the suspended state of the DTS 0 is cleared at time tick 11, DMA transfer of channel 1 starts because there is no currently ongoing DMA transfer and channel 1 is the only channel with a DMA transfer request.

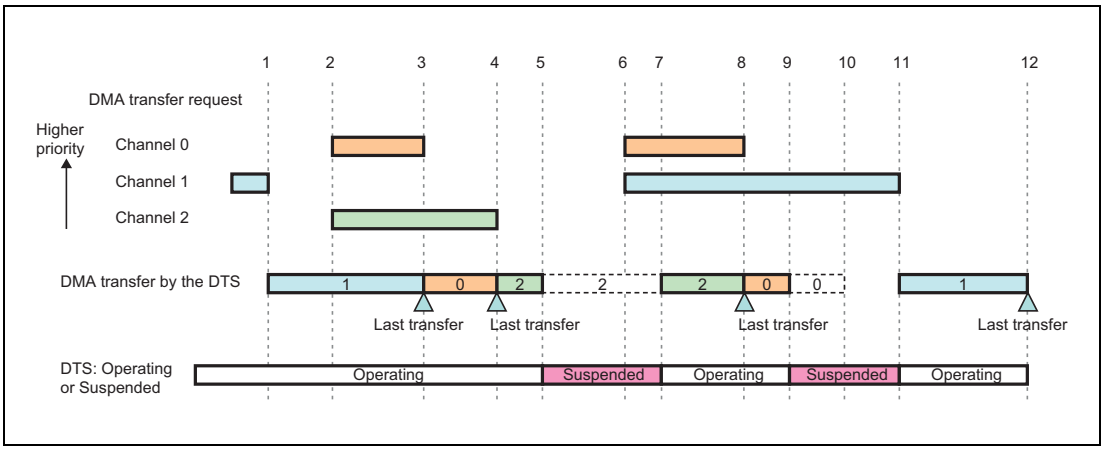



**Figure 7.20 Example of Suspension, Resume, and Transfer Abort of a DTS**

### 7.3.3 Masking and Clearing a Hardware DMA Transfer Request by the DTFR

If a DMAC uses a hardware DMA transfer request, you can temporarily disable (mask) the hardware DMA transfer request output from the DTFR to the DMAC by clearing the hardware DMA transfer source selection enable bit (DTFRn.REQEN) in the DTFR setting register.

Also, if a hardware DMA transfer source is used, you can clear a hardware DMA transfer request retained in the DTFR by using the hardware DMA transfer request clear (DTFRRQCn.DRQC) bit in the DTFR transfer request clear register.

Even if you suspend or abort DMA transfer of a DMAC channel, the hardware DMA transfer request selection/hold circuit of the DTFR is still running, and consequently, the DTFR may retain a hardware DMA transfer request that came to the DTFR during the suspension or transfer abort period of the DMAC channel. When you resume or start DMA transfer of a DMAC channel, clear the hardware DMA transfer request retained in the DTFR as required.

Be careful that if DTFR hardware DMA transfer source selection enable bit is set to disable (DTFRn.REQEN = 0) by software while DMAC is executing block transfer, the ongoing block transfer is suspended. (see **Section 7.2.5.2, Generating and Accepting a Hardware DMA Transfer Request**)

### 7.3.4 Masking and Clearing a Hardware DMA Transfer Request by the DTSFSL

As for a DTS, you can temporarily disable (mask) a DMA transfer request from a channel to the DTS by clearing the transfer request enable bit (DTFSLnnn.REQEN) in the DTSFSL operation setting register. (The masking is actually done by excluding the channel from the candidates in DTS channel arbitration in the DTSFSL.)

Also, you can clear a DMA transfer request retained in the DTSFSL by using the transfer request clear (DTFSCnnn.DRQC) bit in the DTSFSL transfer request clear register.

Regardless of the state of the DTS and the transfer request enable bit (DTFSLnnn.REQEN) of the DTSFSL, the DTSFSL always monitors the hardware transfer source input from outside, and a DMA transfer request for a channel is set when a hardware transfer source for the channel is input to the DTSFSL. When you resume or start DTS transfer, clear the hardware DMA transfer request retained in the DTSFSL as required.

### 7.3.5 List of Suspend, Resume, and Transfer Abort Functions

**Table 7.9 List of Suspend, Resume, and Transfer Abort Functions**

| Function | How to execute the function | Operation | Possibility of DMA transfer abort | Master that can execute the function (See Section 7.5, Reliability Function.) |
|---|---|---|---|---|
| DMA suspension and resume by software control | Setting and clearing the DMACCTL.DMASPD. | All channels are in the suspended state. | Not possible *1 | Special master |
| Suspension and resume of a DMAC channel | Clearing and setting the DCENn.DTE in each channel register.*2 | DMA transfer of a channel is suspended. | Possible (by clearing the DMA transfer request flag during suspension) | Special master, and general master assigned to the channel. |
| Suspension and resume of a DTS | Setting and clearing the DTSCTL1.DTSUST. | DMA transfer of a DTS is suspended. | Possible (by setting the DTSCTL2.DTSTIT during suspension) | Special master |

Note 1. In order to abort DMA transfer, you need to either abort transfer for the DMAC channel or abort transfer for the DTS.

Note 2. In case that the continuous transfer enable bit (DTCTn.MLE) is set, please clear (or set) the continuous transfer enable bit (DTCTn.MLE) first.

## 7.4 Error Control

### 7.4.1 Type of Error

DMA can generate the following two types of errors.

- DMA Transfer Error
  DMA transfer errors are generated by ECC errors or by guard violations.
  Guard errors in the read cycle or write cycle, ECC errors may occur in the read cycle. This error can be generated in all DMAC and DTS channels during execution of DMA transfer.
- DTSRAM Error
  A DTSRAM error is generated when an ECC error is detected by the DTSRAM read access, which is performed by the DTS when fetching the transfer information.

Both DMA Transfer Error types are propagated to the same input of the Error Control Module, DTSRAM Error types are propagated to other inputs. In detail please refer **29.3.1 Error Input** of **Section 29, Error Control Module (ECM)**.

### 7.4.2 DMA Transfer Error

#### 7.4.2.1 Operation of a DMAC When DMA Transfer Error Occurs

When DMA transfer error occurs in a DMAC, the transfer error flag (DCSTn.ER) in the DMAC transfer status register of the channel with the DMA transfer error. The DMAC error register (DMACER) shows the transfer error flags of all 16 DMAC channels.

While the transfer error flag of a channel is set, a new DMA cycle is not executed if the transfer error case DMA transfer disable setting (DTCTn.ESE) bit is set. On the other hand, a DMA cycle is executed regardless of the value of the transfer error flag if the transfer error case DMA transfer disable setting (DTCTn.ESE) bit is cleared.

If you want to abort the DMA transfer of a channel with DMA transfer error, follow the procedure to abort DMA transfer of the DMAC channel.

If DMA transfer error occurs during the read cycle of a DMA cycle, the write cycle is not executed. If DMA transfer error occurs during the write cycle of a DMA cycle, the validity of the write is not guaranteed.

Regardless of whether DMA transfer error occurs in the read cycle or write cycle of a DMA cycle, the source address, destination address, transfer count, and address reload count registers are updated.

#### 7.4.2.2 Operation of a DTS When DMA Transfer Error Occurs

When DMA transfer error occurs in a DTS, the DTS error flag (DTSER1.DTSER) in the DTS error register is set, and the DTS channel number with the DMA transfer error is stored in the DTS error channel (DTSER1.DTSERCH) in the same register.

If DMA transfer error occurs in a single transfer, a TI write back is executed to finish the DMA cycle.

If DMA transfer error occurs in the middle of a block transfer and the transfer error case DMA transfer abort setting (DTTCTnnn.ESE) is set, the remaining DMA cycles in the block transfer are not executed, but a TI write back is executed to finish the DMA cycle. At the same time, the DTS transfer status (DTSSTS.DTSACT) bit in the DTS status register is cleared. If DMA transfer error occurs in the middle of a block transfer and the transfer error case DMA transfer abort setting (DTTCTnnn.ESE) is cleared, the block transfer continues regardless of the DMA transfer error.

If DMA transfer error occurs during the read cycle of a DMA cycle, the write cycle is not executed. If DMA transfer error occurs during the write cycle of a DMA cycle, the validity of the write is not guaranteed.

Regardless of whether DMA transfer error occurs in the read cycle or write cycle of a DMA cycle, the source address, destination address, transfer count, and address reload count registers are updated, and the TI is updated by a TI write back.

If the DTS error flag in the DTS error register is set, a TI fetch is executed when the DTS accepts a DMA transfer request for the channel with the same channel number as the one stored in the DTS error channel. If, as a result of the TI fetch, the transfer error case DMA transfer abort setting (DTTCTnnn.ESE) is found to be set, a DMA cycle and a TI write back are not executed. If the transfer error case DMA transfer abort setting (DTTCTnnn.ESE) is cleared, DMA transfer is executed.

If the DTS error flag in the DTS error register is set, DMA transfer is executed when the DTS accepts a DMA transfer request for a channel with a channel number other than the one stored in the DTS error channel.

### 7.4.3 DTSRAM Error

There are two types of DTSRAM errors detected in the DTSRAM read access: ECC 1-bit error and ECC 2-bit error.

If an ECC 1-bit error is detected during a TI fetch, error corrected data is used, and DMA transfer continues. If an ECC 1-bit error is detected during DTS channel register access from software, error corrected data is returned as read data. In either case, the DTSRAM SEC error flag (DTSER2.RAMSED) in the DTS error register 2 is set, and the address of the error location in the DTSRAM is stored to the DTSRAM SEC error address (DTSER2.RAMSEDAD). In addition, the error is notified to the ECM according to DTSRAM error notification control register (DTRERINT.SEDIE).

If an ECC 2-bit error is detected during a TI fetch, handling of the DMA transfer request is terminated without executing a DMA cycle and TI write back. If an ECC 2-bit error is detected during DTS channel register access from software, peripheral bus error is notified. In either case, the DTSRAM DED error flag (DTSER2.RAMDED) in the DTS error register 2 is set, and the address of the error location in the DTSRAM is stored to the DTSRAM DED error address (DTSER2.RAMDEDAD). In addition, the error is notified to the ECM according to DTSRAM error notification control register (DTRERINT.DEDIE).

#### 7.4.3.1 DTSRAM ECC Test

The proper generation of the error correction code of the DTS RAM can be tested with the following algorithm:

1. Make sure that the RAM location to be tested is not modified by any ongoing DTS transfer.
   It is recommended to disable DTS entirely.
2. Write 0x4000_0002 to DTSRAM Test Control Register (DTRTSCTL).
   Enable ECC Test Mode, encode ECC from write data.
3. Write data to any DTSRAM location.
4. Read data from the same DTSRAM location.
5. Check ECC data in DTSRAM Test Reading Data Register (DTRTRDAT).
6. Restore DTS activity as required.

#### 7.4.3.2 Stimulation of DTSRAM ECC Error

DTSRAM ECC errors can be stimulated to test the proper operation of the error control module. The following algorithm can be used to stimulate such an ECC error:

1. Make sure that the RAM location to be tested is not modified by any ongoing DTS transfer.
   It is recommended to disable DTS entirely.
   The test is intended be done with ECC enabled and 1-bit error correction enabled (default setting of DTRECCTL).
2. Write 0x4000_0003 to DTSRAM Test Control Register (DTRTSCTL).
   Enable ECC Test Mode, write ECC from DTRTWDAT.TWDAT[6:0].
3. Write good or faulty ECC into DTRTWDAT.TWDAT[6:0].
4. Write data to any DTSRAM location.
5. Read data from the same DTSRAM location.
   good ECC should not generate an ECC error.
   faulty ECC should generate an ECC error.
6. Restore DTS activity as required.

## 7.5 Reliability Function

### 7.5.1 Overview

In this product, DMA is a resource used by multiple masters (PE). In order for DMA to support multi-core configuration, the following reliability functions are offered.

- Register access protection function
- Master information inherit function

### 7.5.2 Register Access Protection Function

This product is designed to assign each DMA channel to a PE.

The register access protection function allows access to the transfer information of each DMA channel from the master (PE) assigned to the channel but prohibits access from other masters.

The register access protection function prevents the channel settings from being updated by masters other than the one assigned to the channel. Please note that any master can always read all registers and that the DMAC/DTS controller cannot read or write its own registers.

#### 7.5.2.1 Identifying the Accessing Master

DMA identifies a master based on the ID of the accessing CPU (PEID), the system protection ID configured by the accessing CPU (SPID), and the state of PSW.UM.

#### 7.5.2.2 Special Master Access

The DMA subsystem handles accesses from a special master in UM = 0 as a special master access. In special master access, write access to all DMA registers is allowed. Both PE1 and PE2 can do special master accesses.

#### 7.5.2.3 General Master Access

In master access, access to the following registers is allowed.

- Channel registers of the channels assigned by the channel assignment. (For details, see **Section 7.5.2.4, Channel Assignment**.)

In general master access, write access to registers other than the above is not allowed.

#### 7.5.2.4 Channel Assignment

To each channel, DMA can assign a master (PE) so that the master is allowed to use the channel. Channel assignment is configured in the channel master setting register (DMnnCM in the case of a DMAC and DTSnnnCM in the case of a DTS) by the CPU in the supervisor mode.

In general master access, the master assigned to a channel by channel assignment is allowed to access the channel registers of the channel. If the channel registers of a channel is write accessed by a master other than the master assigned to the channel, the access is called illegal access. For information about illegal access, see **Section 7.5.2.5, Illegal Access**.

#### 7.5.2.5 Illegal Access

DMA handles the following access as illegal access.

(a) General master write access to the global registers

(b) General master write access to the channel registers of a channel by a master other than the master assigned to the channel

DMA's actions against illegal access are as follows.

For both cases (a) and (b),

- Write access is ignored.

Only for the case (b),

- The information about the illegal access is stored in a register access protection violation register.
- The DMAC0, DMAC1, and DTS have their own register access protection violation registers (DM0CMV, DM1CMV, and DTSCMV respectively).
- DMA assert illegal access notification to ECM.

Only the special master can access the register access protection violation registers. The special master can check whether illegal access has occurred by checking the register access protection violation registers periodically or checking DMA illegal access error of ECM.

In addition, it is recommended that, when a master tries to use DMA and configures transfer information in the channel registers, the master should check whether the configuration has been successfully completed without illegal access by, for example, reading back the settings.

### 7.5.3 Master Information Inherit Function

In this product, DMA access inherits master information that is equivalent to the master information of the PE assigned to the DMA channel.

The master information that is output from DMA is as in **Table 7.10**.

**Table 7.10 Master Information That Is Output from DMA**

| Meaning | Value that is output from DMA |
|---|---|
| SPID*[1] | Same as the SPID bit value in the channel master setting register as long as it is between 2 and 31. |
| PEID | Fixed to 4 |

Note 1. In case that register value is 0 or 1, DMA outputs 2 instead of register value

## 7.5.4 Other Reliability Functions

### 7.5.4.1 Restriction on the Next Channel in the Chain

The reliability function limits the channels you can select as the next channel in the chain. When you use the chain function, the channel master settings of a channel and its next channel in the chain must be the same.

The chain function is designed so that a channel and its next channel in the chain are managed by the same master.

When DMA detects that different masters are assigned to a channel and its next channel in the chain, it is deemed illegal and the chain function is suppressed. More specifically, when DMA tries to execute the chain function, DMA compares the chain master settings of the channel and its next channel in the chain, and if the settings are the same for both PEID and UM, the chain function is allowed and a chain request is sent to the next channel. If the settings are not the same for either PEID or UM, a chain request is not sent.

# 7.6 Setting Up DMA Transfer

## 7.6.1 Overview of Setting Up DMA

Table 7.11 Recommended DMA/DTS channel Configuration Sequence

| No. | Master that configures the setting | Description | | Register | Necessity of the setting |
|---|---|---|---|---|---|
| 1 | Special master (CPU in the supervisor mode) | Overall DMA operation setting | DTSPR0 to DTSPR7 | DTS channel priority setting | Mandatory (if a DTS is used) |
| 2 | | | DM00CM to DM17CM | DMAC channel master setting | Mandatory (if a DMAC is used) |
| 3 | | | DTS000CM to DTS127CM | DTS channel master setting | Mandatory (if a DTS is used) |
| 4 | | Status clear | DTSERC | DTS error clear register | Recommended |
| 5 | | | CMVC | Channel protection violation clear register | Recommended |
| 6 | Master assigned to the DMAC channel | Channel setting | DSAn | DMAC source address | Mandatory |
| 7 | | | DDAn | DMAC destination address | Mandatory |
| 8 | | | DTCn | DMAC transfer count | Mandatory |
| 9 | | | DTCTn | DMAC transfer control | Mandatory |
| 10 | | | DRSAn | DMAC reload source address | Mandatory (if the reload function is used) |
| 11 | | | DRDAn | DMAC reload destination address | Mandatory (if the reload function is used) |
| 12 | | | DRTCn | DMAC reload transfer count | Mandatory (if the reload function is used) |
| 13 | | | DTCCn | DMAC transfer count compare | Mandatory (if the transfer count match interrupt is used) |
| 14 | | | DTFRn | DTFR setting register | Mandatory |
| 15 | | Status clear | DCSTCn | DMAC transfer status clear | Mandatory |
| 16 | | | DTFRRQCn | DTFR transfer request clear | Recommended |
| 17 | | Channel operation enable | DCENn | DMAC channel operation enable setting | Mandatory |
| 18 | Master assigned to the DTS channel | Channel setting | DTSAnnn | DTS source address | Mandatory |
| 19 | | | DTDAnnn | DTS destination address | Mandatory |
| 20 | | | DTTCnnn | DTS transfer count | Mandatory |
| 21 | | | DTTCTnnn | DTS transfer control | Mandatory |
| 22 | | | DTRSAnnn | DTS reload source address | Mandatory (if the reload function is used) |
| 23 | | | DTRDAnnn | DTS reload destination address | Mandatory (if the reload function is used) |
| 24 | | | DTRTCnnn | DTS reload transfer count | Mandatory (if the reload function is used) |
| 25 | | | DTTCCnnn | DTS transfer count compare | Mandatory (if the transfer count match interrupt is used) |
| 26 | | Status clear | DTFSCnnn | DTSFSL transfer request clear | Recommended |
| 27 | | Transfer request enable | DTFSLnnn | DTSFSL operation setting | Mandatory |

### 7.6.2 Setting Up the Overall DMA Operation

You need to set up the overall DMA operation before you start using DMA.

To configure the overall DMA operation, the special master (a CPU in the supervisor mode) needs to set up global registers. Global registers can be set up only by special master access. For details, see **Section 7.5, Reliability Function**.

The following registers must be set up to configure the overall DMA operation.

- DTS channel priority setting registers (DTSPRn, n = 0 to 7)
  Those registers configure the priority level of each DTS channel used for DTS channel arbitration.
- DMAC channel master setting registers (DMnnCM)
- DTS channel master setting registers (DTSnnnCM)

Those registers configure channel assignment. (For details, see **Section 7.5, Reliability Function**.)

If the DMAC channel master setting registers and the DTS channel master setting registers are not properly set, DMA channel setting and DMA transfer cannot be executed properly.

Also, if errors are detected in the following registers while the overall DMA operation is set up, clearing the errors is recommended.

- DTS error register 1 (DTSER1)
- DTS error register 2 (DTSER2)
- DMAC0 register access protection violation register (DM0CMV)
- DMAC1 register access protection violation register (DM1CMV)
- DTS register access protection violation register (DTSCMV)

## 7.6.3 Setting Up the DMA Channel Setting

The DMA channel setting defines the transfer information and transfer source for each DMAC and DTS channel.

To configure the DMA channel setting, the master assigned to each channel by the channel assignment needs to set up channel registers.

### 7.6.3.1 Setting Up the DMAC Channel Setting

Follow the procedure below to set up the DMAC channel setting and use the DMAC.

#### (1) Disabling the DMAC Channel Operation

If the channel operation enable (DTE) in the DMAC channel operation enable setting register (DCENn) is set, clear the DTE bit to disable the channel operation.

#### (2) Setting Up the Transfer Information

When you set up the transfer information of the DMAC, the following registers need to be set up.

- DMAC source address register (DSAn)
- DMAC destination address register (DDAn)
- DMAC transfer count register (DTCn)
- DMAC transfer control register (DTCTn)
- DMAC reload source address register (DRSAn)
- DMAC reload destination address register (DRDAn)
- DMAC reload transfer count register (DRTCn)
- DMAC transfer count compare register (DTCCn)

#### (3) Setting Up the DMA Transfer Request

While setting the transfer information, you need to set up the DMA transfer request selection assignment (DTCTn.DRS) bit in the DMAC transfer control register (DTCTn) to define whether the hardware or software DMA transfer request is used.

You cannot use both the hardware and software DMA transfer requests for the same channel at the same time.

If you use the hardware DMA transfer request, you need to select one source used as the hardware DMA transfer request out of 128 hardware DMA transfer sources using the hardware DMA transfer source selection (DTFRn.REQSEL) in the DTFR setting register. Also, you need to enable the hardware DMA transfer source selection (DTFRn.REQEN) in the same register.

The DTFR may retain a hardware DMA transfer request that came before the hardware DMA transfer source is selected. Clear the hardware DMA transfer request (DTFRRQn.DRQ) retained in the DTFR using the DTFR transfer request clear register (DTFRRQCn) if necessary.

If you use the software DMA transfer request, disable the hardware DMA transfer source selection (DTFRn.REQEN) in the DTFR setting register.

#### (4) Clearing the Transfer Status

The DMAC transfer status register (DCSTn) may retain the result of the previous DMA transfer. You need to clear the flags in the DMAC transfer status register using the DMAC transfer status clear register (DCSTCn).

#### (5) Enabling the DMAC Channel Operation

Set the channel operation enable (DCENn.DTE) bit in the DMAC channel operation enable setting register to enable the channel operation.

After the channel operation enable bit is set, the DMAC can accept a DMA transfer request and start DMA transfer.

### 7.6.3.2 Setting Up the DTS Channel Setting

Follow the procedure below to set up the DTS channel setting and use the DTS.

#### (1) Disabling the Transfer Request by the DTSFSL

Clear the transfer request enable (DTFSLnnn.REQEN) bit in the DTSFSL operation setting register of the DTS channel you want to set up the channel setting for. This procedure is not mandatory but recommended in order to prevent a DMA transfer request from being sent mistakenly to the DTS channel currently being configured.

It is also recommended to check the DTS status register (DTSSTS) and confirm that DMA transfer is not ongoing for the DTS channel currently being configured.

#### (2) Setting Up the Transfer Information

When you set up the transfer information of the DTS, the following registers need to be set up to configure the transfer information.

- DTS source address register (DTSAnnn)
- DTS destination address register (DTDAnnn)
- DTS transfer count register (DTTCnnn)
- DTS transfer control register (DTTCTnnn)
- DTS reload source address register (DTRSAnnn)
- DTS reload destination address register (DTRDAnnn)
- DTS reload transfer count register (DTRTCnnn)
- DTS transfer count compare register (DTTCCnnn)

#### (3) Setting Up the DMA Transfer Request

Unlike a DMAC, a DTS does not care whether a DMA transfer request is a hardware DMA transfer request or software DMA transfer request. A DTS has a transfer request pending bit for each channel in the DTSFSL, and both a hardware and software DMA transfer requests are retained in the same transfer request pending bit (DTFSTnnn.DRQ). Therefore, a DTS has no setting for selecting whether the hardware or software transfer request is used.

The DTSFSL may retain a DMA transfer request that came before the transfer information is set up. Clear the DMA transfer request (DTFSTnnn.DRQ) retained in the DTSFSL if necessary, using the DTFSL transfer request clear register (DTFSCnnn).

#### (4) Enabling the Transfer Request by the DTSFSL

Set the transfer request enable (DTFSLnnn.REQEN) bit in the DTSFSL operation setting register to enable the DMA transfer request for the DTS channel.

After the transfer request enable bit for the DTSFSL is set, the DTS can accept a DMA transfer request and start DMA transfer.

# 7.7 DMA Trigger Source

## 7.7.1 List of DMA Trigger Sources

The DMA trigger source assignment for DMA channel n is set in the DTFR setting register (DTFRn).

**Table 7.12 List of DMA Trigger Sources (1/4)**

| | Function/Module | DMA Trigger Source |
|---|---|---|
| DMACTRG[0] | Pin | External Interrupt0 |
| DMACTRG[1] | | External Interrupt1 |
| DMACTRG[2] | | External Interrupt2 |
| DMACTRG[3] | | External Interrupt3 |
| DMACTRG[4] | | External Interrupt4 |
| DMACTRG[5] | | External Interrupt5 |
| DMACTRG[6] | | External Interrupt6 |
| DMACTRG[7] | | External Interrupt7 |
| DMACTRG[8] | | External Interrupt8 (not available for P1M-C device and P1H-C (4MB, BGA-156) device) |
| DMACTRG[9] | | External Interrupt9 (not available for P1M-C (BGA-156) and P1H-C (4MB, BGA-156) device) |
| DMACTRG[10] | GTM | ARU_NEW_DATA0 interrupt |
| DMACTRG[11] | | ARU_NEW_DATA1 interrupt |
| DMACTRG[12] | | ARU_ACC_ACK interrupt |
| DMACTRG[13] | | CMP Shared interrupt |
| DMACTRG[14] | TIM | TIM Shared interrupts (TIM0_IRQ0) |
| DMACTRG[15] | | TIM Shared interrupts (TIM0_IRQ1) |
| DMACTRG[16] | | TIM Shared interrupts (TIM0_IRQ2) |
| DMACTRG[17] | | TIM Shared interrupts (TIM0_IRQ3) |
| DMACTRG[18] | | TIM Shared interrupts (TIM0_IRQ4) |
| DMACTRG[19] | | TIM Shared interrupts (TIM0_IRQ5) |
| DMACTRG[20] | | TIM Shared interrupts (TIM0_IRQ6) |
| DMACTRG[21] | | TIM Shared interrupts (TIM0_IRQ7) |
| DMACTRG[22] | | TIM Shared interrupts (TIM1_IRQ0) |
| DMACTRG[23] | | TIM Shared interrupts (TIM1_IRQ1) |
| DMACTRG[24] | | TIM Shared interrupts (TIM1_IRQ2) |
| DMACTRG[25] | | TIM Shared interrupts (TIM1_IRQ3) |
| DMACTRG[26] | | TIM Shared interrupts (TIM1_IRQ4) |
| DMACTRG[27] | | TIM Shared interrupts (TIM1_IRQ5) |
| DMACTRG[28] | | TIM Shared interrupts (TIM1_IRQ6) |
| DMACTRG[29] | | TIM Shared interrupts (TIM1_IRQ7) |

Table 7.12 List of DMA Trigger Sources (2/4)

| | Function/Module | DMA Trigger Source |
|---|---|---|
| DMACTRG[30] | MCS | MCS Interrupt for channel (MCS0_IRQ0) |
| DMACTRG[31] | | MCS Interrupt for channel (MCS0_IRQ1) |
| DMACTRG[32] | | MCS Interrupt for channel (MCS0_IRQ2) |
| DMACTRG[33] | | MCS Interrupt for channel (MCS0_IRQ3) |
| DMACTRG[34] | | MCS Interrupt for channel (MCS0_IRQ4) |
| DMACTRG[35] | | MCS Interrupt for channel (MCS0_IRQ5) |
| DMACTRG[36] | | MCS Interrupt for channel (MCS0_IRQ6) |
| DMACTRG[37] | | MCS Interrupt for channel (MCS0_IRQ7) |
| DMACTRG[38] | | MCS Interrupt for channel (MCS0_IRQ8) |
| DMACTRG[39] | | MCS Interrupt for channel (MCS1_IRQ0) (not available for P1M-C device) |
| DMACTRG[40] | | MCS Interrupt for channel (MCS1_IRQ1) (not available for P1M-C device) |
| DMACTRG[41] | | MCS Interrupt for channel (MCS1_IRQ2) (not available for P1M-C device) |
| DMACTRG[42] | | MCS Interrupt for channel (MCS1_IRQ3) (not available for P1M-C device) |
| DMACTRG[43] | | MCS Interrupt for channel (MCS1_IRQ4) (not available for P1M-C device) |
| DMACTRG[44] | | MCS Interrupt for channel (MCS1_IRQ5) (not available for P1M-C device) |
| DMACTRG[45] | ATOM | ATOM Shared interrupts (ATOM0_IRQ0) |
| DMACTRG[46] | | ATOM Shared interrupts (ATOM0_IRQ1) |
| DMACTRG[47] | | ATOM Shared interrupts (ATOM0_IRQ2) |
| DMACTRG[48] | | ATOM Shared interrupts (ATOM0_IRQ3) |
| DMACTRG[49] | | ATOM Shared interrupts (ATOM0_IRQ4) |
| DMACTRG[50] | | ATOM Shared interrupts (ATOM0_IRQ5) |
| DMACTRG[51] | | ATOM Shared interrupts (ATOM0_IRQ6) |
| DMACTRG[52] | | ATOM Shared interrupts (ATOM0_IRQ7) |
| DMACTRG[53] | | ATOM Shared interrupts (ATOM1_IRQ0) |
| DMACTRG[54] | | ATOM Shared interrupts (ATOM1_IRQ1) |
| DMACTRG[55] | | ATOM Shared interrupts (ATOM1_IRQ2) |
| DMACTRG[56] | | ATOM Shared interrupts (ATOM1_IRQ3) |
| DMACTRG[57] | | ATOM Shared interrupts (ATOM1_IRQ4) (not available for P1M-C device) |
| DMACTRG[58] | | ATOM Shared interrupts (ATOM1_IRQ5) (not available for P1M-C device) |
| DMACTRG[59] | | ATOM Shared interrupts (ATOM1_IRQ6) (not available for P1M-C device) |
| DMACTRG[60] | | ATOM Shared interrupts (ATOM1_IRQ7) (not available for P1M-C device) |
| DMACTRG[61] | | ATOM Shared interrupts (ATOM2_IRQ0) (not available for P1M-C device) |
| DMACTRG[62] | | ATOM Shared interrupts (ATOM2_IRQ1) (not available for P1M-C device) |
| DMACTRG[63] | | ATOM Shared interrupts (ATOM2_IRQ2) (not available for P1M-C device) |
| DMACTRG[64] | | ATOM Shared interrupts (ATOM2_IRQ3) (not available for P1M-C device) |
| DMACTRG[65] | | ATOM Shared interrupts (ATOM2_IRQ4) (not available for P1M-C device) |
| DMACTRG[66] | ADCF0 | Trigger group for SG0 of ADCF_0 |
| DMACTRG[67] | | Trigger group for SG1 of ADCF_0 |
| DMACTRG[68] | | Trigger group for SG2 of ADCF_0 |
| DMACTRG[69] | | Trigger group for SG3 of ADCF_0 |
| DMACTRG[70] | | Trigger group for SG4 of ADCF_0 |
| DMACTRG[71] | | Multiplex interrupt for ADCF_0 |

Table 7.12 List of DMA Trigger Sources (3/4)

| | Function/Module | DMA Trigger Source |
|---|---|---|
| DMACTRG[72] | ADCF1 | Trigger group for SG0 of ADCF_1 |
| DMACTRG[73] | | Trigger group for SG1 of ADCF_1 |
| DMACTRG[74] | | Trigger group for SG2 of ADCF_1 |
| DMACTRG[75] | | Trigger group for SG3 of ADCF_1 |
| DMACTRG[76] | | Trigger group for SG4 of ADCF_1 |
| DMACTRG[77] | | Multiplex interrupt for ADCF_1 |
| DMACTRG[78] | CSIH | Transmission Interrupt Reception signal for CSIH_0 |
| DMACTRG[79] | | Transmission Interrupt Communication signal for CSIH_0 |
| DMACTRG[80] | | Transmission Interrupt for JOB Completion for CSIH_0 |
| DMACTRG[81] | | Transmission Interrupt Reception signal for CSIH_1 |
| DMACTRG[82] | | Transmission Interrupt Communication signal for CSIH_1 |
| DMACTRG[83] | | Transmission Interrupt for JOB Completion for CSIH_1 |
| DMACTRG[84] | | Transmission Interrupt Reception signal for CSIH_2 |
| DMACTRG[85] | | Transmission Interrupt Communication signal for CSIH_2 |
| DMACTRG[86] | | Transmission Interrupt for JOB Completion for CSIH_2 |
| DMACTRG[87] | | Transmission Interrupt Reception signal for CSIH_3 |
| DMACTRG[88] | | Transmission Interrupt Communication signal for CSIH_3 |
| DMACTRG[89] | | Transmission Interrupt for JOB Completion for CSIH_3 |
| DMACTRG[90] | RLIN | RLIN3_0 interrupt 0 receive |
| DMACTRG[91] | | RLIN3_0 interrupt 0 transmit |
| DMACTRG[92] | | RLIN3_1 interrupt 0 receive |
| DMACTRG[93] | | RLIN3_1 interrupt 0 transmit |
| DMACTRG[94] | | RLIN3_2 interrupt 0 receive (not available for P1M-C device and P1H-C (4MB, BGA-156) device) |
| DMACTRG[95] | | RLIN3_2 interrupt 0 transmit (not available for P1M-C device and P1H-C (4MB, BGA-156) device) |
| DMACTRG[96] | | RLIN3_3 interrupt 0 receive (not available for P1M-C device and P1H-C (4MB, BGA-156) device) |
| DMACTRG[97] | | RLIN3_3 interrupt 0 transmit (not available for P1M-C device and P1H-C (4MB, BGA-156) device) |
| DMACTRG[98] | SENT | receive interrupt for SENT_0 |
| DMACTRG[99] | | receive interrupt for SENT_1 |
| DMACTRG[100] | | receive interrupt for SENT_2 |
| DMACTRG[101] | | receive interrupt for SENT_3 |
| DMACTRG[102] | | receive interrupt for SENT_4 (not available for P1M-C (BGA-156) and P1H-C (4MB, BGA-156) device) |
| DMACTRG[103] | | receive interrupt for SENT_5 (not available for P1M-C (BGA-156) and P1H-C (4MB, BGA-156) device) |
| DMACTRG[104] | | receive interrupt for SENT_6 (not available for P1M-C device and P1H-C (4MB, BGA-156) device) |
| DMACTRG[105] | | receive interrupt for SENT_7 (not available for P1M-C device and P1H-C (4MB, BGA-156) device |
| DMACTRG[106] | | receive interrupt for SENT_8 (not available for P1M-C/P1H-C device) |
| DMACTRG[107] | | receive interrupt for SENT_9 (not available for P1M-C/P1H-C device) |

Table 7.12 List of DMA Trigger Sources (4/4)

| | Function/Module | DMA Trigger Source |
|---|---|---|
| DMACTRG[108] | Reserved | — |
| DMACTRG[109] | Reserved | — |
| DMACTRG[110] | Reserved | — |
| DMACTRG[111] | Reserved | — |
| DMACTRG[112] | Reserved | — |
| DMACTRG[113] | Reserved | — |
| DMACTRG[114] | Reserved | — |
| DMACTRG[115] | Reserved | — |
| DMACTRG[116] | Reserved | — |
| DMACTRG[117] | Reserved | — |
| DMACTRG[118] | Reserved | — |
| DMACTRG[119] | Reserved | — |
| DMACTRG[120] | Reserved | — |
| DMACTRG[121] | Reserved | — |
| DMACTRG[122] | Reserved | — |
| DMACTRG[123] | Reserved | — |
| DMACTRG[124] | Reserved | — |
| DMACTRG[125] | Reserved | — |
| DMACTRG[126] | Reserved | — |
| DMACTRG[127] | Reserved | — |

# 7.8 DTS Trigger Source

## 7.8.1 List of DTS Trigger Sources

**Table 7.13** shows the DTS trigger source assignment for DTS channel n.

Table 7.13 List of DTS Trigger Sources (1/5)

| Channel | Function/Module | Primary DTS Trigger Source | Secondary DTS Trigger Source |
|---|---|---|---|
| DTSTRG[0] | Pin | External Interrupt 0 | none |
| DTSTRG[1] | | External Interrupt 1 | none |
| DTSTRG[2] | | External Interrupt 2 | none |
| DTSTRG[3] | | External Interrupt 3 | none |
| DTSTRG[4] | | External Interrupt 4 | STM0_0 |
| DTSTRG[5] | | External Interrupt 5 | STM0_1 |
| DTSTRG[6] | | External Interrupt 6 | STM0_2 |
| DTSTRG[7] | | External Interrupt 7 | STM0_3 |
| DTSTRG[8] | | External Interrupt 8 (not available for P1M-C device and P1H-C (4MB, BGA-156) device) | STM0_4 |
| DTSTRG[9] | | External Interrupt 9 (not available for P1M-C (BGA-156) and P1H-C (4MB, BGA-156) device) | STM0_5 |
| DTSTRG[10] | TIM | TIM Shared interrupts (TIM0_IRQ0) | none |
| DTSTRG[11] | | TIM Shared interrupts (TIM0_IRQ1) | none |
| DTSTRG[12] | | TIM Shared interrupts (TIM0_IRQ2) | none |
| DTSTRG[13] | | TIM Shared interrupts (TIM0_IRQ3) | none |
| DTSTRG[14] | | TIM Shared interrupts (TIM0_IRQ4) | none |
| DTSTRG[15] | | TIM Shared interrupts (TIM0_IRQ5) | none |
| DTSTRG[16] | | TIM Shared interrupts (TIM0_IRQ6) | none |
| DTSTRG[17] | | TIM Shared interrupts (TIM0_IRQ7) | none |
| DTSTRG[18] | | TIM Shared interrupts (TIM1_IRQ0) | none |
| DTSTRG[19] | | TIM Shared interrupts (TIM1_IRQ1) | none |
| DTSTRG[20] | | TIM Shared interrupts (TIM1_IRQ2) | none |
| DTSTRG[21] | | TIM Shared interrupts (TIM1_IRQ3) | none |
| DTSTRG[22] | | TIM Shared interrupts (TIM1_IRQ4) | STM1_0 (not available for P1M-C device) |
| DTSTRG[23] | | TIM Shared interrupts (TIM1_IRQ5) | STM1_1 (not available for P1M-C device) |
| DTSTRG[24] | | TIM Shared interrupts (TIM1_IRQ6) | STM1_2 (not available for P1M-C device) |
| DTSTRG[25] | | TIM Shared interrupts (TIM1_IRQ7) | STM1_3 (not available for P1M-C device) |
| DTSTRG[26] | ADCF0 | Trigger group for SG0 of ADCF_0 | none |
| DTSTRG[27] | | Trigger group for SG1 of ADCF_0 | none |
| DTSTRG[28] | | Trigger group for SG2 of ADCF_0 | none |
| DTSTRG[29] | | Trigger group for SG3 of ADCF_0 | none |
| DTSTRG[30] | | Trigger group for SG4 of ADCF_0 | none |

Table 7.13 List of DTS Trigger Sources (2/5)

| Channel | Function/Module | Primary DTS Trigger Source | Secondary DTS Trigger Source |
|---|---|---|---|
| DTSTRG[31] | ADCF1 | Trigger group for SG0 of ADCF_1 | none |
| DTSTRG[32] | | Trigger group for SG1 of ADCF_1 | none |
| DTSTRG[33] | | Trigger group for SG2 of ADCF_1 | none |
| DTSTRG[34] | | Trigger group for SG3 of ADCF_1 | none |
| DTSTRG[35] | | Trigger group for SG4 of ADCF_1 | none |
| DTSTRG[36] | CSIH | DTS trigger for received register of CSIH_0 #TG0 | none |
| DTSTRG[37] | | DTS trigger for transmit register of CSIH_0 #TG0 | none |
| DTSTRG[38] | | DTS trigger for received register of CSIH_0 #TG1 | none |
| DTSTRG[39] | | DTS trigger for transmit register of CSIH_0 #TG1 | none |
| DTSTRG[40] | | DTS trigger for received register of CSIH_0 #TG2 | none |
| DTSTRG[41] | | DTS trigger for transmit register of CSIH_0 #TG2 | none |
| DTSTRG[42] | | DTS trigger for received register of CSIH_0 #TG3 | none |
| DTSTRG[43] | | DTS trigger for transmit register of CSIH_0 #TG3 | none |
| DTSTRG[44] | | DTS trigger for received register of CSIH_0 #TG4 | none |
| DTSTRG[45] | | DTS trigger for transmit register of CSIH_0 #TG4 | none |
| DTSTRG[46] | | DTS trigger for received register of CSIH_0 #TG5 | none |
| DTSTRG[47] | | DTS trigger for transmit register of CSIH_0 #TG5 | none |
| DTSTRG[48] | | DTS trigger for received register of CSIH_0 #TG6 | none |
| DTSTRG[49] | | DTS trigger for transmit register of CSIH_0 #TG6 | none |
| DTSTRG[50] | | DTS trigger for received register of CSIH_0 #TG7 | none |
| DTSTRG[51] | | DTS trigger for transmit register of CSIH_0 #TG7 | none |
| DTSTRG[52] | | DTS trigger for received register of CSIH_1 #TG0 | none |
| DTSTRG[53] | | DTS trigger for transmit register of CSIH_1 #TG0 | none |
| DTSTRG[54] | | DTS trigger for received register of CSIH_1 #TG1 | none |
| DTSTRG[55] | | DTS trigger for transmit register of CSIH_1 #TG1 | none |
| DTSTRG[56] | | DTS trigger for received register of CSIH_1 #TG2 | none |
| DTSTRG[57] | | DTS trigger for transmit register of CSIH_1 #TG2 | none |
| DTSTRG[58] | | DTS trigger for received register of CSIH_1 #TG3 | none |
| DTSTRG[59] | | DTS trigger for transmit register of CSIH_1 #TG3 | none |
| DTSTRG[60] | | DTS trigger for received register of CSIH_1 #TG4 | none |
| DTSTRG[61] | | DTS trigger for transmit register of CSIH_1 #TG4 | none |
| DTSTRG[62] | | DTS trigger for received register of CSIH_1 #TG5 | none |
| DTSTRG[63] | | DTS trigger for transmit register of CSIH_1 #TG5 | none |
| DTSTRG[64] | | DTS trigger for received register of CSIH_1 #TG6 | none |
| DTSTRG[65] | | DTS trigger for transmit register of CSIH_1 #TG6 | none |
| DTSTRG[66] | | DTS trigger for received register of CSIH_1 #TG7 | ATOM Shared interrupts (ATOM2_IRQ3) (not available for P1M-C device) |
| DTSTRG[67] | | DTS trigger for transmit register of CSIH_1 #TG7 | ATOM Shared interrupts (ATOM2_IRQ4) (not available for P1M-C device) |

Table 7.13 List of DTS Trigger Sources (3/5)

| Channel | Function/Module | Primary DTS Trigger Source | Secondary DTS Trigger Source |
|---|---|---|---|
| DTSTRG[68] | CSIH | DTS trigger for received register of CSIH_2 #TG0 | none |
| DTSTRG[69] | | DTS trigger for transmit register of CSIH_2 #TG0 | none |
| DTSTRG[70] | | DTS trigger for received register of CSIH_2 #TG1 | none |
| DTSTRG[71] | | DTS trigger for transmit register of CSIH_2 #TG1 | none |
| DTSTRG[72] | | DTS trigger for received register of CSIH_2 #TG2 | none |
| DTSTRG[73] | | DTS trigger for transmit register of CSIH_2 #TG2 | none |
| DTSTRG[74] | | DTS trigger for received register of CSIH_2 #TG3 | none |
| DTSTRG[75] | | DTS trigger for transmit register of CSIH_2 #TG3 | none |
| DTSTRG[76] | | DTS trigger for received register of CSIH_2 #TG4 | none |
| DTSTRG[77] | | DTS trigger for transmit register of CSIH_2 #TG4 | none |
| DTSTRG[78] | | DTS trigger for received register of CSIH_2 #TG5 | none |
| DTSTRG[79] | | DTS trigger for transmit register of CSIH_2 #TG5 | none |
| DTSTRG[80] | | DTS trigger for received register of CSIH_2 #TG6 | ATOM Shared interrupts (ATOM1_IRQ7) (not available for P1M-C device) |
| DTSTRG[81] | | DTS trigger for transmit register of CSIH_2 #TG6 | ATOM Shared interrupts (ATOM2_IRQ0) (not available for P1M-C device) |
| DTSTRG[82] | | DTS trigger for received register of CSIH_2 #TG7 | ATOM Shared interrupts (ATOM2_IRQ1) (not available for P1M-C device) |
| DTSTRG[83] | | DTS trigger for transmit register of CSIH_2 #TG7 | ATOM Shared interrupts (ATOM2_IRQ2) (not available for P1M-C device) |
| DTSTRG[84] | | DTS trigger for received register of CSIH_3 #TG0 | none |
| DTSTRG[85] | | DTS trigger for transmit register of CSIH_3 #TG0 | none |
| DTSTRG[86] | | DTS trigger for received register of CSIH_3 #TG1 | none |
| DTSTRG[87] | | DTS trigger for transmit register of CSIH_3 #TG1 | none |
| DTSTRG[88] | | DTS trigger for received register of CSIH_3 #TG2 | none |
| DTSTRG[89] | | DTS trigger for transmit register of CSIH_3 #TG2 | none |
| DTSTRG[90] | | DTS trigger for received register of CSIH_3 #TG3 | none |
| DTSTRG[91] | | DTS trigger for transmit register of CSIH_3 #TG3 | none |
| DTSTRG[92] | | DTS trigger for received register of CSIH_3 #TG4 | none |
| DTSTRG[93] | | DTS trigger for transmit register of CSIH_3 #TG4 | none |
| DTSTRG[94] | | DTS trigger for received register of CSIH_3 #TG5 | none |
| DTSTRG[95] | | DTS trigger for transmit register of CSIH_3 #TG5 | none |
| DTSTRG[96] | | DTS trigger for received register of CSIH_3 #TG6 | ATOM Shared interrupts (ATOM1_IRQ3) |
| DTSTRG[97] | | DTS trigger for transmit register of CSIH_3 #TG6 | ATOM Shared interrupts (ATOM1_IRQ4) (not available for P1M-C device) |
| DTSTRG[98] | | DTS trigger for received register of CSIH_3 #TG7 | ATOM Shared interrupts (ATOM1_IRQ5) (not available for P1M-C device) |
| DTSTRG[99] | | DTS trigger for transmit register of CSIH_3 #TG7 | ATOM Shared interrupts (ATOM1_IRQ6) (not available for P1M-C device) |

Table 7.13 List of DTS Trigger Sources (4/5)

| Channel | Function/Module | Primary DTS Trigger Source | Secondary DTS Trigger Source |
|---|---|---|---|
| DTSTRG[100] | RLIN3 | RLIN3_0 interrupt 0 receive | MCS Interrupt for channel (MCS1_IRQ0) (not available for P1M-C device) |
| DTSTRG[101] | | RLIN3_0 interrupt 0 transmit | MCS Interrupt for channel (MCS1_IRQ1) (not available for P1M-C device) |
| DTSTRG[102] | | RLIN3_1 interrupt 0 receive | MCS Interrupt for channel (MCS1_IRQ2) (not available for P1M-C device) |
| DTSTRG[103] | | RLIN3_1 interrupt 0 transmit | MCS Interrupt for channel (MCS1_IRQ3) (not available for P1M-C device) |
| DTSTRG[104] | | RLIN3_2 interrupt 0 receive (not available for P1M-C device and P1H-C (4MB, BGA-156) device) | MCS Interrupt for channel (MCS1_IRQ4) (not available for P1M-C device) |
| DTSTRG[105] | | RLIN3_2 interrupt 0 transmit (not available for P1M-C device and P1H-C (4MB, BGA-156) device) | MCS Interrupt for channel (MCS1_IRQ5) (not available for P1M-C device) |
| DTSTRG[106] | | RLIN3_3 interrupt 0 receive (not available for P1M-C device and P1H-C (4MB, BGA-156) device) | STM0_6 |
| DTSTRG[107] | | RLIN3_3 interrupt 0 transmit (not available for P1M-C device and P1H-C (4MB, BGA-156) device) | STM0_7 |
| DTSTRG[108] | SENT | receive interrupt for SENT_0 | none |
| DTSTRG[109] | | receive interrupt for SENT_1 | none |
| DTSTRG[110] | | receive interrupt for SENT_2 | none |
| DTSTRG[111] | | receive interrupt for SENT_3 | none |
| DTSTRG[112] | | receive interrupt for SENT_4 (not available for P1M-C (BGA-156) and P1H-C (4MB, BGA-156) device) | none |
| DTSTRG[113] | | receive interrupt for SENT_5 (not available for P1M-C (BGA-156) and P1H-C (4MB, BGA-156) device) | none |
| DTSTRG[114] | | receive interrupt for SENT_6 (not available for P1M-C device and P1H-C (4MB, BGA-156) device) | STM1_4 (not available for P1M-C device) |
| DTSTRG[115] | | receive interrupt for SENT_7 (not available for P1M-C device and P1H-C (4MB, BGA-156) device) | STM1_5 (not available for P1M-C device) |
| DTSTRG[116] | | receive interrupt for SENT_8 (not available for P1M-C/P1H-C device) | STM1_6 (not available for P1M-C device) |
| DTSTRG[117] | | receive interrupt for SENT_9 (not available for P1M-C/P1H-C device) | STM1_7 (not available for P1M-C device) |

Table 7.13 List of DTS Trigger Sources (5/5)

| Channel | Function/Module | Primary DTS Trigger Source | Secondary DTS Trigger Source |
|---|---|---|---|
| DTSTRG[118] | MCS | MCS Interrupt for channel (MCS0_IRQ0) | ATOM Shared interrupts (ATOM0_IRQ1) |
| DTSTRG[119] | | MCS Interrupt for channel (MCS0_IRQ1) | ATOM Shared interrupts (ATOM0_IRQ2) |
| DTSTRG[120] | | MCS Interrupt for channel (MCS0_IRQ2) | ATOM Shared interrupts (ATOM0_IRQ3) |
| DTSTRG[121] | | MCS Interrupt for channel (MCS0_IRQ3) | ATOM Shared interrupts (ATOM0_IRQ4) |
| DTSTRG[122] | | MCS Interrupt for channel (MCS0_IRQ4) | ATOM Shared interrupts (ATOM0_IRQ5) |
| DTSTRG[123] | | MCS Interrupt for channel (MCS0_IRQ5) | ATOM Shared interrupts (ATOM0_IRQ6) |
| DTSTRG[124] | | MCS Interrupt for channel (MCS0_IRQ6) | ATOM Shared interrupts (ATOM0_IRQ7) |
| DTSTRG[125] | | MCS Interrupt for channel (MCS0_IRQ7) | ATOM Shared interrupts (ATOM1_IRQ0) |
| DTSTRG[126] | | MCS Interrupt for channel (MCS0_IRQ8) | ATOM Shared interrupts (ATOM1_IRQ1) |
| DTSTRG[127] | | ATOM Shared interrupts (ATOM0_IRQ0) | ATOM Shared interrupts (ATOM1_IRQ2) |

## 7.9 Global Register

### 7.9.1 List of Global Register Addresses

Address = Base address “FFFF 8000h” + Offset address

**Table 7.14 List of Global Register Addresses (1/2)**

| Offset Address | Register Symbol | Meaning | Accessed Permission | |
|---|---|---|---|---|
| | | | Special Master | General Master |
| $0000_H$ | DMACTL | DMA control register | √ | × |
| $0010_H$ | DTSCTL1 | DTS control register 1 | √ | × |
| $0014_H$ | DTSCTL2 | DTS control register 2 | √ | × |
| $0018_H$ | DTSSTS | DTS status register | √ | × |
| $0020_H$ | DMACER | DMAC error register | √ | × |
| $0024_H$ | DTSER1 | DTS error register 1 | √ | × |
| $0028_H$ | DTSER2 | DTS error register 2 | √ | × |
| $002C_H$ | DTSERC | DTS error clear register | √ | × |
| $0030_H$ | DM0CMV | DMAC0 register access protection violation register | √ | × |
| $0034_H$ | DM1CMV | DMAC1 register access protection violation | √ | × |
| $0038_H$ | DTSCMV | DTS register access protection violation register | √ | × |
| $003C_H$ | CMVC | Register access protection violation clear register | √ | × |
| $004C_H$ | TFRSTS | Transfer status register | √ | × |
| $0060_H$ | DTSPR0 | DTS channel priority setting 0 | √ | × |
| $0064_H$ | DTSPR1 | DTS channel priority setting 1 | √ | × |
| $0068_H$ | DTSPR2 | DTS channel priority setting 2 | √ | × |
| $006C_H$ | DTSPR3 | DTS channel priority setting 3 | √ | × |
| $0070_H$ | DTSPR4 | DTS channel priority setting 4 | √ | × |
| $0074_H$ | DTSPR5 | DTS channel priority setting 5 | √ | × |
| $0078_H$ | DTSPR6 | DTS channel priority setting 6 | √ | × |
| $007C_H$ | DTSPR7 | DTS channel priority setting 7 | √ | × |
| $0080_H$ | DTRECCTL | DTSRAM ECC control register | √ | × |
| $0084_H$ | DTRERINT | DTSRAM Error notification control register | √ | × |
| $0094_H$ | DTRTSCTL | DTSRAM test control register | √ | × |
| $0098_H$ | DTRTWDAT | DTSRAM test write data register | √ | × |
| $009C_H$ | DTRTRDAT | DTSRAM test read data register | √ | × |
| $00A0_H$ | ADECCTCL | ECConBUS address ECC test control register | √ | × |
| $00A4_H$ | ADECCTDT | ECConBUS address ECC test data register | √ | × |
| $0100_H$ | DM00CM | DMAC0 channel 0 channel master setting | √ | × |
| $0104_H$ | DM01CM | DMAC0 channel 1 channel master setting | √ | × |
| $0108_H$ | DM02CM | DMAC0 channel 2 channel master setting | √ | × |
| $010C_H$ | DM03CM | DMAC0 channel 3 channel master setting | √ | × |
| $0110_H$ | DM04CM | DMAC0 channel 4 channel master setting | √ | × |
| $0114_H$ | DM05CM | DMAC0 channel 5 channel master setting | √ | × |
| $0118_H$ | DM06CM | DMAC0 channel 6 channel master setting | √ | × |
| $011C_H$ | DM07CM | DMAC0 channel 7 channel master setting | √ | × |
| $0120_H$ | DM10CM | DMAC1 channel 0 channel master setting | √ | × |

Table 7.14 List of Global Register Addresses (2/2)

| Offset Address | Register Symbol | Meaning | Accessed Permission | |
|---|---|---|---|---|
| | | | Special Master | General Master |
| $0124_H$ | DM11CM | DMAC1 channel 1 channel master setting | √ | × |
| $0128_H$ | DM12CM | DMAC1 channel 2 channel master setting | √ | × |
| $012C_H$ | DM13CM | DMAC1 channel 3 channel master setting | √ | × |
| $0130_H$ | DM14CM | DMAC1 channel 4 channel master setting | √ | × |
| $0134_H$ | DM15CM | DMAC1 channel 5 channel master setting | √ | × |
| $0138_H$ | DM16CM | DMAC1 channel 6 channel master setting | √ | × |
| $013C_H$ | DM17CM | DMAC1 channel 7 channel master setting | √ | × |
| $0200_H$ + 4 × [DTS channel number] *1 ($0200_H$ to $03FC_H$) | DTSnnnCM *1 | DTS channel nnn channel master setting *1 | √ | × |

Note 1. [DTS channel number] and “nnn” in the register symbols and meanings are numbers in the range from 000 to 127.

**Note:** √: supported
x: not supported

## 7.9.2 Details of Global Registers

### 7.9.2.1 DMACTL — DMA Control Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8000_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DMA SPD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 7.15 DMACTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | DMASPD | DMA suspension<br>This bit shows whether DMA transfer for all channels is suspended. If a user writes 1 to this bit, DMA transfer for all channels can be suspended. If a user writes 0 to this bit, suspension of DMA transfer for all channels can be cleared. The suspension controlled by this bit is independent from the suspension controlled by the transfer enable bit (DTE) of each DMAC channel and the suspension setting bit (DTSUST) for a DTS. That means, if this bit is 1, all DMA transfers are suspended regardless of the values of the DTE bit of each channel and the DTSUST bit of the DTS.<br>Writing to this bit does not affect the DTE bit of each channel and the DTSUST bit of the DTS.<br>0: DMA suspension cleared<br>1: DMA suspension request/DMA suspension ongoing |

### 7.9.2.2 DTSCTL1 — DTS Control Register 1

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8010_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DTSUST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 7.16 DTSCTL1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | DTSUST | DTS suspension<br>This bit shows whether DMA transfer of a DTS is suspended. If a user writes 1 to this bit, DMA transfer of a DTS can be suspended.<br>0: DTS suspension cleared<br>1: DTS suspension request/DTS suspension ongoing |

### 7.9.2.3 DTSCTL2 — DTS Control Register 2

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8014_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DTSTIT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 7.17 DTSCTL2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | DTSTIT | DTS transfer abort request<br>While the DTS is suspended, a user can write 1 to this bit to abort the suspended DMA transfer.<br>When the suspended DMA transfer of a DTS is aborted, the DTSSTS.DTSACT bit is cleared to 0.<br>The read value of this bit is always zero. |

### 7.9.2.4 DTSSTS — DTS Status Register

**Access:** This register can be read in 32-bit units.

**Address:** FFFF 8018$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DTS CYC | DTSA CH6 | DTSA CH5 | DTSA CH4 | DTSA CH3 | DTSA CH2 | DTSA CH1 | DTSA CH0 | DTSA CT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.18 DTSSTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is read. |
| 8 | DTSCYC | DMA cycle execution state<br>This bit shows whether a DMA cycle is ongoing in the DTS.<br>0: DMA cycle is not ongoing.<br>1: DMA cycle is ongoing.<br>After suspending DTS transfers, DTSCYC may be polled to assure that any possibly ongoing DTS cycle has finished |
| 7 to 1 | DTSACH[6:0] | DTS transfer channel<br>If there is a channel in the DTS executing DMA transfer, the channel number is shown.<br>If there is no channel in the DTS executing DMA transfer, the channel number of the last DMA transfer is shown. |
| 0 | DTSACT | DTS transfer status<br>This bit shows whether there is a channel in the DTS executing DMA transfer.<br>0: There is a channel in the DTS executing DMA transfer.<br>1: There is no channel in the DTS executing DMA transfer.<br>DTSACT is asserted during TI fetch, all DMA cycles and TI write-back. It is cleared after the last cycle of the TI write back is completed.<br>If the DTS is put into the suspended state while there is a channel executing DMA transfer, this bit remains 1.<br>If a DTS transfer abort request is made using the DTSCTL2.DTSTIT bit, the suspended DTS transfer is aborted, and this bit is cleared to 0.<br>When DMA transfer error occurs and the DMA transfer is aborted, this bit is cleared. |

### 7.9.2.5 DMACER — DMAC Error Register

**Access:** This register can be read in 32-bit units.

**Address:** FFFF 8020$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DM1 ER7 | DM1 ER6 | DM1 ER5 | DM1 ER4 | DM1 ER3 | DM1 ER2 | DM1 ER1 | DM1 ER0 | DM0 ER7 | DM0 ER6 | DM0 ER5 | DM0 ER4 | DM0 ER3 | DM0 ER2 | DM0 ER1 | DM0 ER0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.19 DMACER Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. |
| 15 to 8 | DM1ER[7:0] | DMAC1 DMA transfer error status<br>These bits show the DMA transfer error status of channels 0 through 7 of the DMAC1.<br>Each bit is mapped from the DCSTn.ER bit of each channel of the DMAC1 and is read-only.<br>0: DMA transfer error is not generated<br>1: DMA transfer error is generated |
| 7 to 0 | DM0ER[7:0] | DMAC0 DMA transfer error status<br>These bits show the DMA transfer error status of channels 0 through 7 of the DMAC0.<br>Each bit is mapped from the DCSTn.ER bit of each channel of the DMAC0 and is read-only.<br>0: DMA transfer error is not generated<br>1: DMA transfer error is generated |

### 7.9.2.6 DTSER1 — DTS Error Register 1

**Access:** This register can be read in 32-bit units.

**Address:** FFFF $8024_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | DTSERCH6 | DTSERCH5 | DTSERCH4 | DTSERCH3 | DTSERCH2 | DTSERCH1 | DTSERCH0 | — | — | — | — | — | — | DTSERWR | DTSER |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.20 DTSER1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 15 | Reserved | When read, the value after reset is read. |
| 14 to 8 | DTSERCH[6:0] | DTS error channel<br>These bits show the DTS channel number of the first DMA transfer error after the DTSER bit is cleared to 0.<br>These bits are read-only and cannot be cleared. |
| 7 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DTSERWR | DTS DMA transfer error occurring cycle<br>This bit is updated at the same time as setting of the DTS DMA transfer error flag (DTSER), indicating which cycle of read or write the DMA transfer error occurs in. This bit is not updated when a new DMA transfer error occurs after the DTSER bit has been set. If the DTSER bit is cleared, this bit is also cleared to 0.<br>0: DMA transfer error occurs in read cycle.<br>1: DMA transfer error occurs in write cycle. |
| 0 | DTSER | DTS DMA transfer error flag<br>This bit shows whether DMA transfer error is generated in the DTS.<br>0: DMA transfer error is not generated<br>1: DMA transfer error is generated<br>If DMA transfer error is generated in the DTS while this bit is 0, this bit is set, and DTSERCH6 to 0 retains the DTS channel number of the DMA transfer error.<br>If DMA transfer error is generated in the DTS while this bit is 1, this bit remains 1, and DTSERCH6 to 0 does not change.<br>This bit can be cleared by using the DTSERC register. |

### 7.9.2.7 DTSER2 — DTS Error Register 2

**Access:** This register can be read in 32-bit units.

**Address:** FFFF $8028_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RAMDED | RAMDEDOV | — | — | RAMDEDAD11 | RAMDEDAD10 | RAMDEDAD9 | RAMDEDAD8 | RAMDEDAD7 | RAMDEDAD6 | RAMDEDAD5 | RAMDEDAD4 | RAMDEDAD3 | RAMDEDAD2 | RAMDEDAD1 | RAMDEDAD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RAMSED | RAMSEDOV | — | — | RAMSEDAD11 | RAMSEDAD10 | RAMSEDAD9 | RAMSEDAD8 | RAMSEDAD7 | RAMSEDAD6 | RAMSEDAD5 | RAMSEDAD4 | RAMSEDAD3 | RAMSEDAD2 | RAMSEDAD1 | RAMSEDAD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.21 DTSER2 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | RAMDED | DTSRAM DED error flag<br>This bit shows whether the DED error is generated in the read access to the DTSRAM.<br>0: DED error is not generated in the DTSRAM<br>1: DED error is generated in the DTSRAM<br>If DED error is generated in the DTSRAM while this bit is 0, this bit is set, and RAMDEDAD11 to 0 retains the DTSRAM address of the error.<br>If DED error is generated in the DTSRAM while this bit is 1, this bit remains 1, and RAMDEDAD11-0 does not change.<br>This bit can be cleared by using the DTSERC register. |
| 30 | RAMDEDOV | DTSRAM DED error overflow flag<br>This bit is set when the RAMDED bit is 1 and the DED error occurs in DTSRAM read access whose address is different from that specified by the RAMDEDAD11 to 0 bit. This bit can be cleared by operation of the DTSERC register. |
| 29, 28 | Reserved | When read, the value after reset is read. |
| 27 to 16 | RAMDEDAD [11:0] | DTSRAM DED error address<br>These bits show the DTSRAM address of the first DTSRAM DED error after the RAMDED bit is cleared to 0.<br>These bits are read-only and cannot be cleared. |
| 15 | RAMSED | DTSRAM SEC error flag<br>This bit shows whether the SEC error is generated in the read access to the DTSRAM.<br>0: SEC error is not generated in the DTSRAM<br>1: SEC error is generated in the DTSRAM<br>If SEC error is generated in the DTSRAM while this bit is 0, this bit is set, and RAMSEDAD11 to 0 retains the DTSRAM address of the error.<br>If SEC error is generated in the DTSRAM while this bit is 1, this bit remains 1, and RAMSEDAD11 to 0 does not change.<br>This bit can be cleared by using the DTSERC register. |
| 14 | RAMSEDOV | DTSRAM SEC error overflow flag<br>This bit is set when the RAMSED bit is 1 and the SEC error occurs in DTSRAM read access whose address is different from that specified by the RAMSEDAD11 to 0 bit. This bit can be cleared by operation of the DTSERC register. |
| 13, 12 | Reserved | When read, the value after reset is read. |

**Table 7.21 DTSER2 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 to 0 | RAMSEDAD [11:0] | DTSRAM SEC error address<br>These bits show the DTSRAM address of the first DTSRAM SEC error after the RAMSED bit is cleared to 0.<br>These bits are read-only and cannot be cleared. |

### 7.9.2.8 DTSERC — DTS Error Clear Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 802C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RAMDEDC | RAMDEDOVC | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RAMSEDC | RAMSEDOVC | — | — | — | — | — | — | — | — | — | — | — | — | — | DTSERC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 7.22 DTSERC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | RAMDEDC | DTSRAM DED error flag clear<br>If a user writes 1 to this bit, the DTSRAM DED error flag (DTSER2.RAMDED) is cleared.<br>0 is always read from this bit. |
| 30 | RAMDEDOVC | DTSRAM DED error overflow flag clear<br>When the user writes 1 to this bit, the DTSRAM DED error overflow flag (DTSER2.RAMDEDOV) is cleared. The read value of this bit is always 0. |
| 29 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 | RAMSEDC | DTSRAM SEC error flag clear<br>If a user writes 1 to this bit, the DTSRAM SEC error flag (DTSER2.RAMSED) is cleared.<br>0 is always read from this bit. |
| 14 | RAMSEDOVC | DTSRAM DED error overflow flag clear<br>When the user writes 1 to this bit, the DTSRAM DED error overflow flag (DTSER2.RAMSEDOV) is cleared. The read value of this bit is always 0. |
| 13 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | DTSERC | DTS error flag clear<br>If a user writes 1 to this bit, the DTS DMA error flag (DTSER1.DTSER) is cleared.<br>0 is always read from this bit. |

### 7.9.2.9 DM0CMV — DMAC0 Register Access Protection Violation Register

**Access:** This register can be read in 32-bit units.

**Address:** FFFF 8030$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID2 | PEID1 | PEID0 | — | — | — | — | — | — | SPID4 | SPID3 | SPID2 | SPID1 | SPID0 | UM | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | VCH2 | VCH1 | VCH0 | — | — | — | VF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.23 DM0CMV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29<br>22 to 18<br>17 | PEID[2:0]<br>SPID[4:0]<br>UM | Illegal access master information<br>These bits show the accessing master information of the first illegal access after the VF bit is cleared to 0.<br>If illegal access occurs while the VF bit is 1, these bits do not change.<br>These bits are read-only and cannot be cleared. |
| 28 to 23<br>16 to 7 | Reserved | When read, the value after reset is read. |
| 6 to 4 | VCH[2:0] | Illegal access channel<br>These bits show the channel number (0 to 7) of the first illegal access after the VF bit is cleared to 0.<br>If illegal access occurs while the VF bit is 1, these bits do not change.<br>These bits are read-only and cannot be cleared. |
| 3 to 1 | Reserved | When read, the value after reset is read. |
| 0 | VF | Illegal access flag<br>This bit shows whether illegal access occurs in the DMAC0.<br>0: No illegal access has occurred in the DMAC0<br>1: Illegal access has occurred in the DMAC0<br>If illegal access occurs in the DMAC0 while this bit is 0, this bit is set, and PEID, SPID, UM and VCH store their respective information.<br>If illegal access occurs in the DMAC0 while this bit is 1, this bit remains 1, and PEID, SPID, UM and VCH do not change.<br>This bit can be cleared by using the CMVC register. |

### 7.9.2.10 DM1CMV — DMAC1 Register Access Protection Violation Register

**Access:** This register can be read in 32-bit units.

**Address:** FFFF $8034_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID2 | PEID1 | PEID0 | — | — | — | — | — | — | SPID4 | SPID3 | SPID2 | SPID1 | SPID0 | UM | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | VCH2 | VCH1 | VCH0 | — | — | — | VF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.24 DM1CMV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29<br>22 to 18<br>17 | PEID[2:0]<br>SPID[4:0]<br>UM | Illegal access master information<br>These bits show the accessing master information of the first illegal access after the VF bit is cleared to 0.<br>If illegal access occurs while the VF bit is 1, these bits do not change.<br>These bits are read-only and cannot be cleared. |
| 28 to 23<br>16 to 7 | Reserved | When read, the value after reset is read. |
| 6 to 4 | VCH[2:0] | Illegal access channel<br>These bits show the channel number (0 to 7) of the first illegal access after the VF bit is cleared to 0.<br>If illegal access occurs while the VF bit is 1, these bits do not change.<br>These bits are read-only and cannot be cleared. |
| 3 to 1 | Reserved | When read, the value after reset is read. |
| 0 | VF | Illegal access flag<br>This bit shows whether illegal access occurs in the DMAC1.<br>0: No illegal access has occurred in the DMAC1<br>1: Illegal access has occurred in the DMAC1<br>If illegal access occurs in the DMAC1 while this bit is 0, this bit is set, and PEID, SPID, UM and VCH store their respective information.<br>If illegal access occurs in the DMAC1 while this bit is 1, this bit remains 1, and PEID, SPID, UM and VCH do not change.<br>This bit can be cleared by using the CMVC register. |

### 7.9.2.11 DTSCMV — DTS Register Access Protection Violation Register

**Access:** This register can be read in 32-bit units.

**Address:** FFFF 8038$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID2 | PEID1 | PEID0 | — | — | — | — | — | — | SPID4 | SPID3 | SPID2 | SPID1 | SPID0 | UM | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | VCH6 | VCH5 | VCH4 | VCH3 | VCH2 | VCH1 | VCH0 | — | — | — | VF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.25 DTSCMV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29<br>22 to 18<br>17 | PEID[2:0]<br>SPID[4:0]<br>UM | Illegal access master information<br>These bits show the accessing master information of the first illegal access after the VF bit is cleared to 0. If illegal access occurs while the VF bit is 1, these bits do not change. These bits are read-only and cannot be cleared. |
| 28 to 23<br>16 to 11 | Reserved | When read, the value after reset is read. |
| 10 to 4 | VCH[6:0] | Illegal access channel<br>These bits show the channel number (0 to 127) of the first illegal access after the VF bit is cleared to 0.<br>If illegal access occurs while the VF bit is 1, these bits do not change.<br>These bits are read-only and cannot be cleared. |
| 3 to 1 | Reserved | When read, the value after reset is read. |
| 0 | VF | Illegal access flag<br>This bit shows whether illegal access occurs in the DTS.<br>0: No illegal access has occurred in the DTS<br>1: Illegal access has occurred in the DTS<br>If illegal access occurs in the DTS while this bit is 0, this bit is set, and PEID, SPID, UM and VCH store their respective information.<br>If illegal access occurs in the DTS while this bit is 1, this bit remains 1, and PEID, SPID, UM and VCH do not change.<br>This bit can be cleared by using the CMVC register. |

### 7.9.2.12 CMVC — Register Access Protection Violation Clear Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 803C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | DTSVC | DM1VC | DM0VC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 7.26 CMVC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 2 | DTSVC | DTS illegal access flag clear<br>The DTS illegal access flag (DTSCMV.VF) can be cleared by writing 1 to this bit. 0 is always read from this bit. |
| 1 | DM1VC | DMAC1 illegal access flag clear<br>The DMAC1 illegal access flag (DM1CMV.VF) can be cleared by writing 1 to this bit. 0 is always read from this bit. |
| 0 | DM0VC | DMAC0 illegal access flag clear<br>The DMAC0 illegal access flag (DM0CMV.VF) can be cleared by writing 1 to this bit. 0 is always read from this bit. |

### 7.9.2.13 TFRSTS — Transfer Status Register

**Access:** This register can be read in 32-bit units.

**Address:** FFFF 804C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DTS CYC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA CY17 | DMA CY16 | DMA CY15 | DMA CY14 | DMA CY13 | DMA CY12 | DMA CY11 | DMA CY10 | DMA CY07 | DMA CY06 | DMA CY05 | DMA CY04 | DMA CY03 | DMA CY02 | DMA CY01 | DMA CY00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.27 TFRSTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | When read, the value after reset is read. |
| 16 | DTSCYC | DMA cycle state of DTS IP block:<br>0: No DTS-DMA cycle executing<br>1: DTS-DMA cycle is executing<br>After suspending DTS transfers, DTSCYC may be polled to assure that any possibly ongoing DTS cycle has finished.<br>This flag is a copy of DTSSTS.DTSCYC |
| 15 to 8<br>7 to 0 | DMACY[17:10]<br>DMACY[07:00] | DMA cycle execution state<br>This bit shows whether a DMA cycle is ongoing in this channel.<br>0: DMA cycle is not ongoing.<br>1: DMA cycle is ongoing.<br>After suspending a DMA channel, CY may be polled to assure that any possibly ongoing DMA cycle has finished.<br>These flags are copies from DCSTn.CY. |

### 7.9.2.14 DTSPRn — DTS Channel Priority Setting (n = 0 to 7)

- DTSPR0

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 8060$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS15 PR1 | DTS15 PR0 | DTS14 PR1 | DTS14 PR0 | DTS13 PR1 | DTS13 PR0 | DTS12 PR1 | DTS12 PR0 | DTS11 PR1 | DTS11 PR0 | DTS10 PR1 | DTS10 PR0 | DTS9 PR1 | DTS9 PR0 | DTS8 PR1 | DTS8 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS7 PR1 | DTS7 PR0 | DTS6 PR1 | DTS6 PR0 | DTS5 PR1 | DTS5 PR0 | DTS4 PR1 | DTS4 PR0 | DTS3 PR1 | DTS3 PR0 | DTS2 PR1 | DTS2 PR0 | DTS1 PR1 | DTS1 PR0 | DTS0 PR1 | DTS0 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.28 DTSPR0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTS[15:0] PR[1:0] | DTS channel [15:0] priority setting<br>These bits configure the priority level of each DTS channel used for DTS channel arbitration.<br>00 is the highest priority, and 11 is the lowest. |

- DTSPR1

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 8064$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS31 PR1 | DTS31 PR0 | DTS30 PR1 | DTS30 PR0 | DTS29 PR1 | DTS29 PR0 | DTS28 PR1 | DTS28 PR0 | DTS27 PR1 | DTS27 PR0 | DTS26 PR1 | DTS26 PR0 | DTS25 PR1 | DTS25 PR0 | DTS24 PR1 | DTS24 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS23 PR1 | DTS23 PR0 | DTS22 PR1 | DTS22 PR0 | DTS21 PR1 | DTS21 PR0 | DTS20 PR1 | DTS20 PR0 | DTS19 PR1 | DTS19 PR0 | DTS18 PR1 | DTS18 PR0 | DTS17 PR1 | DTS17 PR0 | DTS16 PR1 | DTS16 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.29 DTSPR1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTS[31:16] PR[1:0] | DTS channel [31:16] priority setting<br>These bits configure the priority level of each DTS channel used for DTS channel arbitration.<br>00 is the highest priority, and 11 is the lowest. |

- DTSPR2

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8068_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS47 PR1 | DTS47 PR0 | DTS46 PR1 | DTS46 PR0 | DTS45 PR1 | DTS45 PR0 | DTS44 PR1 | DTS44 PR0 | DTS43 PR1 | DTS43 PR0 | DTS42 PR1 | DTS42 PR0 | DTS41 PR1 | DTS41 PR0 | DTS40 PR1 | DTS40 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS39 PR1 | DTS39 PR0 | DTS38 PR1 | DTS38 PR0 | DTS37 PR1 | DTS37 PR0 | DTS36 PR1 | DTS36 PR0 | DTS35 PR1 | DTS35 PR0 | DTS34 PR1 | DTS34 PR0 | DTS33 PR1 | DTS33 PR0 | DTS32 PR1 | DTS32 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.30 DTSPR2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTS[47:32] PR[1:0] | DTS channel [47:32] priority setting<br>These bits configure the priority level of each DTS channel used for DTS channel arbitration.<br>00 is the highest priority, and 11 is the lowest. |

- DTSPR3

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $806C_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS63 PR1 | DTS63 PR0 | DTS62 PR1 | DTS62 PR0 | DTS61 PR1 | DTS61 PR0 | DTS60 PR1 | DTS60 PR0 | DTS59 PR1 | DTS59 PR0 | DTS58 PR1 | DTS58 PR0 | DTS57 PR1 | DTS57 PR0 | DTS56 PR1 | DTS56 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS55 PR1 | DTS55 PR0 | DTS54 PR1 | DTS54 PR0 | DTS53 PR1 | DTS53 PR0 | DTS52 PR1 | DTS52 PR0 | DTS51 PR1 | DTS51 PR0 | DTS50 PR1 | DTS50 PR0 | DTS49 PR1 | DTS49 PR0 | DTS48 PR1 | DTS48 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.31 DTSPR3 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTS[63:48] PR[1:0] | DTS channel [63:48] priority setting<br>These bits configure the priority level of each DTS channel used for DTS channel arbitration.<br>00 is the highest priority, and 11 is the lowest. |

- DTSPR4

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 8070$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS79 PR1 | DTS79 PR0 | DTS78 PR1 | DTS78 PR0 | DTS77 PR1 | DTS77 PR0 | DTS76 PR1 | DTS76 PR0 | DTS75 PR1 | DTS75 PR0 | DTS74 PR1 | DTS74 PR0 | DTS73 PR1 | DTS73 PR0 | DTS72 PR1 | DTS72 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS71 PR1 | DTS71 PR0 | DTS70 PR1 | DTS70 PR0 | DTS69 PR1 | DTS69 PR0 | DTS68 PR1 | DTS68 PR0 | DTS67 PR1 | DTS67 PR0 | DTS66 PR1 | DTS66 PR0 | DTS65 PR1 | DTS65 PR0 | DTS64 PR1 | DTS64 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.32 DTSPR4 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTS[79:64] PR[1:0] | DTS channel [79:64] priority setting<br>These bits configure the priority level of each DTS channel used for DTS channel arbitration.<br>00 is the highest priority, and 11 is the lowest. |

- DTSPR5

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 8074$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS95 PR1 | DTS95 PR0 | DTS94 PR1 | DTS94 PR0 | DTS93 PR1 | DTS93 PR0 | DTS92 PR1 | DTS92 PR0 | DTS91 PR1 | DTS91 PR0 | DTS90 PR1 | DTS90 PR0 | DTS89 PR1 | DTS89 PR0 | DTS88 PR1 | DTS88 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS87 PR1 | DTS87 PR0 | DTS86 PR1 | DTS86 PR0 | DTS85 PR1 | DTS85 PR0 | DTS84 PR1 | DTS84 PR0 | DTS83 PR1 | DTS83 PR0 | DTS82 PR1 | DTS82 PR0 | DTS81 PR1 | DTS81 PR0 | DTS80 PR1 | DTS80 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.33 DTSPR5 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTS[95:80] PR[1:0] | DTS channel [95:80] priority setting<br>These bits configure the priority level of each DTS channel used for DTS channel arbitration.<br>00 is the highest priority, and 11 is the lowest. |

- DTSPR6

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 8078$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS111 PR1 | DTS111 PR0 | DTS110 PR1 | DTS110 PR0 | DTS109 PR1 | DTS109 PR0 | DTS108 PR1 | DTS108 PR0 | DTS107 PR1 | DTS107 PR0 | DTS106 PR1 | DTS106 PR0 | DTS105 PR1 | DTS105 PR0 | DTS104 PR1 | DTS104 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS103 PR1 | DTS103 PR0 | DTS102 PR1 | DTS102 PR0 | DTS101 PR1 | DTS101 PR0 | DTS100 PR1 | DTS100 PR0 | DTS99 PR1 | DTS99 PR0 | DTS98 PR1 | DTS98 PR0 | DTS97 PR1 | DTS97 PR0 | DTS96 PR1 | DTS96 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.34 DTSPR6 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTS[111:96] PR[1:0] | DTS channel [111:96] priority setting<br>These bits configure the priority level of each DTS channel used for DTS channel arbitration.<br>00 is the highest priority, and 11 is the lowest. |

- DTSPR7

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 807C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS127 PR1 | DTS127 PR0 | DTS126 PR1 | DTS126 PR0 | DTS125 PR1 | DTS125 PR0 | DTS124 PR1 | DTS124 PR0 | DTS123 PR1 | DTS123 PR0 | DTS122 PR1 | DTS122 PR0 | DTS121 PR1 | DTS121 PR0 | DTS120 PR1 | DTS120 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTS119 PR1 | DTS119 PR0 | DTS118 PR1 | DTS118 PR0 | DTS117 PR1 | DTS117 PR0 | DTS116 PR1 | DTS116 PR0 | DTS115 PR1 | DTS115 PR0 | DTS114 PR1 | DTS114 PR0 | DTS113 PR1 | DTS113 PR0 | DTS112 PR1 | DTS112 PR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.35 DTSPR7 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTS[127:112] PR[1:0] | DTS channel [127:112] priority setting<br>These bits configure the priority level of each DTS channel used for DTS channel arbitration.<br>00 is the highest priority, and 11 is the lowest. |

### 7.9.2.15 DTRECCTL — DTSRAM ECC Control Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8080_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 7.36 DTRECCTL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | PROT[1:0] | This bits enable or disable writing to the ECCDIS and SECDIS bits.<br>The written data is not retained. The read value is always 0.<br>This bit should be written when (PROT1, PROT0) = (0, 1) |
| 29 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SECDIS | DTSRAM 1-bit error correction disable<br>This bit enables or disables 1-bit error correction when the ECCDIS bit is 0.<br>Set (PROT1, PROT0) = (0, 1) at the same time as writing to this bit.<br>0: Error correction is enabled when the 1-bit error is detected.<br>1: Error correction is disabled when the 1-bit error is detected. |
| 0 | ECCDIS | DTSRAM ECC disable<br>This bit enables or disables DTSRAM ECC error detection and correction.<br>Set (PROT1, PROT0) = (0, 1) at the same time as writing to this bit.<br>0: Error detection and correction are enabled.<br>1: Error detection and correction are disabled.<br>The encoding function is effective when error detection and correction are disabled |

### 7.9.2.16 DTRERINT — DTSRAM Error Notification Control Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8084_H$

**Value after reset:** 0000 $0002_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 7.37 DTRERINT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | DTSRAM DED error external notification enable<br>This bit enables or disables notification of DED error detection to the ECM when the ECCDIS bit in DTRECCTL is 0.<br>0: Notification of DED error to ECM is disabled.<br>1: Notification of DED error to ECM is enabled. |
| 0 | SEDIE | DTSRAM 1-bit error external notification enable<br>This bit enables or disables notification of 1-bit error detection to the ECM when the ECCDIS bit in DTRECCTL is 0.<br>0: Notification of 1-bit error to ECM is disabled.<br>1: Notification of 1-bit error to ECM is enabled. |

### 7.9.2.17 DTRTSCTL — DTSRAM Test Control Register

This register is used for ECC test (self-diagnosis). It enables setting of ECC test mode and selection of ECC data to be written to the DTSRAM.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 8094$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ECCTST | DATSEL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 7.38 DTRTSCTL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | PROT[1:0] | This bits enable or disable writing to the ECCTST and DATSEL bits.<br>The written data is not retained. The read value is always 0.<br>This bit should be written when (PROT1, PROT0) = (0, 1) |
| 29 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | ECCTST | DTSRAM ECC Test Mode<br>This bit enables or disables DTSRAM ECC test mode.<br>Set (PROT1, PROT0) = (0, 1) at the same time as writing to this bit.<br>0: ECC test mode is disabled.<br>1: ECC test mode is enabled. |
| 0 | DATSEL | ECC Test Data Selection<br>This bit is valid when ECCTST is 1 and selects ECC data to be written to the DTSRAM.<br>Set (PROT1, PROT0) = (0, 1) at the same time as writing to this bit.<br>0: ECC encoded from the written data is used.<br>1: The value specified by the DTSRAM test data writing register (DTRTWDAT) is used. |

### 7.9.2.18 DTRTWDAT — DTSRAM Test Write Data Register

This register is used for ECC test (self-diagnosis). It specifies ECC data to be written to the DTSRAM after ECC test mode is enabled (ECCTST = 1).

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 8098$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | TWDAT6 | TWDAT5 | TWDAT4 | TWDAT3 | TWDAT2 | TWDAT1 | TWDAT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.39 DTRTWDAT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 to 0 | TWDAT[6:0] | ECC Test Write Data<br>This bits specifies ECC data to be written to the DTSRAM when DTRTSCTL.ECCTST = 1 and DTRTSCTL.DATSEL = 1.<br>Writing to this bit is enabled when DTRTSCTL.ECCTST = 1<br>When DTRTSCTL.ECCTST = 0, this bit cannot be written and its read value is 0. |

### 7.9.2.19 DTRTRDAT — DTSRAM Test Read Data Register

This register is used for ECC test (self-diagnosis). It reads out ECC data of the DTSRAM after ECC test mode is enabled (ECCTST = 1).

**Access:** This register can be read in 32-bit units.

**Address:** FFFF 809C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | TRDAT 6 | TRDAT 5 | TRDAT 4 | TRDAT 3 | TRDAT 2 | TRDAT 1 | TRDAT 0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.40 DTRTRDAT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is read. |
| 6 to 0 | TRDAT[6:0] | ECC Test Read Data<br>This bits retains the last ECC data read out from the DTSRAM when DTRTSCTL.ECCTST = 1.<br>When DTRTSCTL.ECCTST = 0, the read value of this bit is 0. |

### 7.9.2.20 ADECCTCL — ECC on BUS Address ECC Test Control Register

See **Section 28.2.9.6**.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 80A0$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ECCTST | RWSEL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 7.41 ADECCTCL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | PROT[1:0] | Write control bits of ECCTST bit and RWSEL bit.<br>Write data to these bit are not kept internally. 0 is always read from these bits. |
| 29 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | ECCTST | Address ECC Test Mode<br>Configures ECConBUS address ECC test mode.<br>To update this bit, (PROT1, PROT0) = (0, 1) must be written at the same time.<br>0: Address ECC test mode is disabled.<br>1: Address ECC test mode is enabled.<br>Address ECC output is replaced with the value of ADECCTDT register. |
| 0 | RWSEL | Address ECC Test Cycle Selection<br>Selects either read cycle or write cycle to test address ECC. This bit is valid only at ECCTST=1.<br>To update this bit, (PROT1, PROT0) = (0, 1) must be written at the same time.<br>0: Address ECC output is replaced with test data at DMA read cycle.<br>1: Address ECC output is replaced with test data at DMA write cycle. |

### 7.9.2.21 ADECCTDT — ECC on BUS Address ECC Test Data Register

See **Section 28.2.9.6**.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 80A4$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | ECCDAT6 | ECCDAT5 | ECCDAT4 | ECCDAT3 | ECCDAT2 | ECCDAT1 | ECCDAT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.42 ADECCTDT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 to 0 | ECCDAT[6:0] | ECC test data<br>Specifies ECC test data used at ADECCTCL.ECCTST = 1. |

### 7.9.2.22 DMnnCM — DMAC Channel Master Setting (nn = 00 to 07, 10 to 17)

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8100_H$ + 4 × Ch. No. n (n = 0 to 7)
FFFF $8120_H$ + 4 × Ch. No. n - 10 (n = 10 to 17)

**Value after reset:** 0000 $2008_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID2 | PEID1 | PEID0 | — | — | — | — | — | — | SPID4 | SPID3 | SPID2 | SPID1 | SPID0 | UM | — |
| Value after reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.43 DMnnCM Register Contents**

| Bit Position | Bit Name | Function ion |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 13 | PEID[2:0] | Channel master PEID setting<br>Specifies the PEID information of the master assigned to the channel. |
| 12 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 to 2 | SPID[4:0] | Channel master SPID setting<br>Specifies the SPID information used by the master assigned to the channel.<br>SPID = 0 and 1 are reserved and must not be used.<br>If SPID = 0 or 1 is set, DMA will use SPID = 2. |
| 1 | UM | Channel master UM setting<br>Specifies the UM information of the master assigned to the channel. |
| 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

**CAUTION**

**DM00CM to DM07CM configure the channel master information of the DMAC0 channel 0 to 7 respectively.**

**DM10CM to DM17CM configure the channel master information of the DMAC1 channel 0 to 7 respectively.**

For information about the functions this register offers, see **Section 7.5, Reliability Function**.

### 7.9.2.23 DTSnnnCM — DTS Channel Master Setting Register (nnn = 000 to 127)

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8200_H$ + $4_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID2 | PEID1 | PEID0 | — | — | — | — | — | — | SPID4 | SPID3 | SPID2 | SPID1 | SPID0 | UM | — |
| Value after reset | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMC15 | CMC14 | CMC13 | CMC12 | CMC11 | CMC10 | CMC9 | CMC8 | CMC7 | CMC6 | CMC5 | CMC4 | CMC3 | CM2C | CMC1 | CMC0 |
| Value after reset | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed | Unfixed |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.44 DTS Channel Master Setting Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | PEID[2:0] | Channel master PEID setting Specifies the PEID information of the master assigned to the channel. |
| 28 to 23 | Reserved | When writing, write the value “0”. When read, the unfixed value after reset and “0” after write. |
| 22 to18 | SPID[4:0] | Channel master SPID setting<br>Specifies the SPID information used by the master assigned to the channel.<br>SPID = 0 and 1 are reserved and must not be used.<br>If SPID = 0 or 1 is set, DMA will use SPID = 2. |
| 17 | UM | Channel master UM setting<br>Specifies the UM information of the master assigned to the channel. |
| 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | CMC[15:0] | Transfer count compare<br>In terms of contents, this field is the same as the bit [15:0] of the **Section 7.11.3.8, DTTCCnnn — DTS Transfer Count Compare Register**. |

**CAUTION**

**DTS000CM to DTS127CM configure the channel master information of the DTS channel 0 to 127 respectively.**

For information about the functions this register offers, see **Section 7.5, Reliability Function**.

**CAUTION**

**The lowest 16 bits of this register are shared with the DTS transfer count compare register, one of the DTS channel registers.**
**If you write to this register, the DTS transfer count compare register is updated as well.**

Recommended setup procedure of the DTS channel master setting register

When the special master configures the overall DTS operation, the channel master setting must be configured in the bits 31 to 29, 22 to 16 in this register, and 0 must be specified in the bits 28 to 23,15 to 0.

The bits 28 to 23 of this register are reserved, but you can read and write those bits.

# 7.10 DMAC Channel Register

## 7.10.1 DMAC Channel Register Address

Address = Base address "FFFF $8000_H$" + Offset address

**Table 7.45 DMAC Channel Register Address**

| Offset Address | Register Symbol | Meaning | Accessed Permission*2 | |
|---|---|---|---|---|
| | | | Special Master | General Master |
| $0400_H + 40_H \times$ [channel number]*1 | DSAn*1 | DMAC source address | √ | √ |
| $0404_H + 40_H \times$ [channel number]*1 | DDAn*1 | DMAC destination address | √ | √ |
| $0408_H + 40_H \times$ [channel number]*1 | DTCn*1 | DMAC transfer count | √ | √ |
| $040C_H + 40_H \times$ [channel number]*1 | DTCTn*1 | DMAC transfer control | √ | √ |
| $0410_H + 40_H \times$ [channel number]*1 | DRSAn*1 | DMAC reload source address | √ | √ |
| $0414_H + 40_H \times$ [channel number]*1 | DRDAn*1 | DMAC reload destination address | √ | √ |
| $0418_H + 40_H \times$ [channel number]*1 | DRTCn*1 | DMAC reload transfer count | √ | √ |
| $041C_H + 40_H \times$ [channel number]*1 | DTCCn*1 | DMAC transfer count compare | √ | √ |
| $0420_H + 40_H \times$ [channel number]*1 | DCENn*1 | DMAC channel operation enable setting | √ | √ |
| $0424_H + 40_H \times$ [channel number]*1 | DCSTn*1 | DMAC transfer status | √ | √ |
| $0428_H + 40_H \times$ [channel number]*1 | DCSTSn*1 | DMAC transfer status set | √ | √ |
| $042C_H + 40_H \times$ [channel number]*1 | DCSTCn*1 | DMAC transfer status clear | √ | √ |
| $0430_H + 40_H \times$ [channel number]*1 | DTFRn*1 | DTFR setting | √ | √ |
| $0434_H + 40_H \times$ [channel number]*1 | DTFRRQn*1 | DTFR transfer request status | √ | √ |
| $0438_H + 40_H \times$ [channel number]*1 | DTFRRQCn*1 | DTFR transfer request clear | √ | √ |

Note 1. The [channel number] in the offset addresses and "n" in the register symbols are numbers in the range from 0 to 15, and the correspondence between the channel number n and the channel is as follows.

Note 2. √: supported
x: not supported

| Channel number n | Channel |
|---|---|
| 0 | DMAC0 channel 0 |
| 1 | DMAC0 channel 1 |
| 2 | DMAC0 channel 2 |
| 3 | DMAC0 channel 3 |
| 4 | DMAC0 channel 4 |
| 5 | DMAC0 channel 5 |
| 6 | DMAC0 channel 6 |
| 7 | DMAC0 channel 7 |
| 8 | DMAC1 channel 0 |
| 9 | DMAC1 channel 1 |
| 10 | DMAC1 channel 2 |
| 11 | DMAC1 channel 3 |
| 12 | DMAC1 channel 4 |
| 13 | DMAC1 channel 5 |
| 14 | DMAC1 channel 6 |
| 15 | DMAC1 channel 7 |

## 7.10.2 Details of DMAC Channel Registers

The “n” in the register symbols indicates the DMA channel number (n = 0 to 15).

### 7.10.2.1 DSAn — DMAC Source Address Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8400_H$ + $40_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SA31 | SA30 | SA29 | SA28 | SA27 | SA26 | SA25 | SA24 | SA23 | SA22 | SA21 | SA20 | SA19 | SA18 | SA17 | SA16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SA15 | SA14 | SA13 | SA12 | SA11 | SA10 | SA9 | SA8 | SA7 | SA6 | SA5 | SA4 | SA3 | SA2 | SA1 | SA0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.46 DSAn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SA[31:0] | Source address<br>Specifies the DMA transfer source address. Those bits are updated whenever a DMA cycle is executed. If you read from those bits, the transfer source address for the next DMA cycle is read. |

**CAUTIONS**

1. **It is forbidden to write to those bits when the channel operation is enabled (DTE bit = 1). If you do, the correct operation is not guaranteed.**
2. **The address must be set up while the DTE bit is 0.**
3. **DMA transfer for misaligned data is not supported. The possible lowest four bits of the address for each transfer data size is as follows. (× denotes an arbitrary one bit.)**
   **The correct operation is not guaranteed if you select other setting.**

| Data Size | SA3 | SA2 | SA1 | SA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 7.10.2.2 DDAn — DMAC Destination Address Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8404_H$ + $40_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DA31 | DA30 | DA29 | DA28 | DA27 | DA26 | DA25 | DA24 | DA23 | DA22 | DA21 | DA20 | DA19 | DA18 | DA17 | DA16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DA15 | DA14 | DA13 | DA12 | DA11 | DA10 | DA9 | DA8 | DA7 | DA6 | DA5 | DA4 | DA3 | DA2 | DA1 | DA0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.47 DDAn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DA[31:0] | Destination address<br>Specifies the DMA transfer destination address. Those bits are updated whenever a DMA cycle is executed. If you read from those bits, the transfer destination address for the next DMA cycle is read. |

**CAUTIONS**

1. **It is forbidden to write to those bits when the channel operation is enabled (DTE bit = 1). If you do, the correct operation is not guaranteed.**
2. **The address must be set up while the DTE bit is 0.**
3. **If transfer error is generated in the read cycle of DMA transfer, the write cycle is not executed, but the destination address is updated.**
4. **DMA transfer for misaligned data is not supported. The possible lowest four bits of the address for each transfer data size is as follows. (× denotes an arbitrary one bit.)**
   **The correct operation is not guaranteed if you select other setting.**

| Data Size | DA3 | DA2 | DA1 | DA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 7.10.2.3 DTCn — DMAC Transfer Count Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8408_H$ + $40_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARC15 | ARC14 | ARC13 | ARC12 | ARC11 | ARC10 | ARC9 | ARC8 | ARC7 | ARC6 | ARC5 | ARC4 | ARC3 | ARC2 | ARC1 | ARC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRC15 | TRC14 | TRC13 | TRC12 | TRC11 | TRC10 | TRC9 | TRC8 | TRC7 | TRC6 | TRC5 | TRC4 | TRC3 | TRC2 | TRC1 | TRC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.48 DTCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | ARC[15:0] | Address reload count<br>Specifies the number of transfers until the address reload when the reload function 2 is used, and also specifies the number of transfers when the block transfer 2 is used. If you read from those bits during DMA transfer, the address reload count for the next DMA cycle is read.<br>When the reload function 2 or block transfer 2 is used, ARC[15:0] is decremented by one for every DMA cycle. When the reload function 2 or block transfer 2 is not used, ARC[15:0] is not updated.<br>If the value is $0000_H$, it means that the number of transfers until the address reload when the reload function 2 is used and the number of transfers when the block transfer 2 is used are 65536. |
| 15 to 0 | TRC[15:0] | Transfer count<br>Configures the number of transfers. TRC[15:0] is decremented by one whenever a DMA cycle is executed. If you read from those bits, the remaining number of transfers for the next DMA cycle is read. If the reload function is not used, after the last transfer is complete, the value at the completion ($0000_H$) is retained. |

| TRC15 to 0 | Operation |
|---|---|
| $0000_H$ | The number of transfers is 65536, or the transfer is complete. |
| $0001_H$ | The number of transfers or remaining transfers is 1. |
| : | : |
| $FFFF_H$ | The number of transfers or remaining transfers is 65535. |

**CAUTIONS**

1. **It is forbidden to write to those bits when the channel operation is enabled (DTE bit = 1). If you do, the correct operation is not guaranteed.**
2. **If transfer error is generated in the read cycle of DMA transfer, the write cycle is not executed, but the transfer count and the address reload count are updated.**

### 7.10.2.4 DTCTn — DMAC Transfer Control Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 840C$_H$ + 40$_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ESE | DRS | — | — | — | — | — | CHNSEL2 | CHNSEL1 | SHNSEL0 | CHNE1 | CHNE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CCE | TCE | MLE | RLD2M1 | RLD2M0 | RLD1M1 | RLD1M0 | DACM1 | DACM0 | SACM1 | SACM0 | DS2 | DS1 | DS0 | TRM1 | TRM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.49 DTCTn Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 27 | ESE | Transfer error case DMA transfer disable setting<br>Configures whether a DMA cycle is executed when the DCSTn.ER bit is set due to DMA transfer error.<br>If this bit is cleared to 0, even when the DCSTn.ER bit is set due to DMA transfer error, the following DMA cycles can be executed. If this bit is set to 1, the following DMA cycles are not executed when the DCSTn.ER bit is set due to DMA transfer error.<br>0: DMA cycles are executed while the DCSTn.ER bit is set.<br>1: DMA cycles are not executed while the DCSTn.ER bit is set. |
| 26 | DRS | DMA transfer request selection assignment<br>Selects the type of DMA transfer requests to be accepted.<br>0: Software DMA transfer request<br>1: Hardware DMA transfer request |
| 25 to 21 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 20 to 18 | CHNSEL[2:0] | Next channel in the chain<br>Specifies the next channel in the chain.<br>The next channel must be another channel in the same DMAC. You cannot specify a channel in the different DMAC or in the DTS. You cannot specify the same channel for the next channel. (If you do, the correct operation is not guaranteed.) |
| 17, 16 | CHNE1, 0 | Chain enable<br>Selects the chain function.<br>00: Disabled<br>01: Chain at the last transfer<br>A chain request is generated at the completion of the DMA cycle in which the remaining transfer count is one.<br>10: (Forbidden. No guarantee of operation)<br>11: Always chain<br>A chain request is generated at the completion of every DMA cycle. |
| 15 | CCE | Transfer count match interrupt enable<br>If this bit is set, a transfer count match interrupt is generated at the completion of the DMA cycle in which the transfer count compare register and the remaining transfer count have the same value. |
| 14 | TCE | Transfer completion interrupt enable<br>If this bit is set, a transfer completion interrupt is generated at the completion of the last transfer. |

**Table 7.49 DTCTn Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 13 | MLE | Continuous transfer enable<br>If this bit is set, the DTE bit is not cleared at the completion of DMA transfer. In addition, even if the TC bit is not cleared, DMA transfer starts when there is a DMA transfer request.<br>0: The DTE bit is cleared at the completion of DMA transfer. In addition, the next DMA transfer can start only after the TC bit is cleared.<br>1: The DTE bit is not cleared at the completion of DMA transfer. In addition, even if the TC bit is not cleared, DMA transfer starts when there is a DMA transfer request. |
| 12, 11 | RLD2M1, 0 | Reload function 2 setting<br>Configures the reload function 2.<br>00: Reload function 2 is disabled.<br>01: Reload function 2 is enabled.<br>The source address and address reload count are reloaded at the completion of the DMA cycle in which the address reload count is 1.<br>10: Reload function 2 is enabled.<br>The destination address and address reload count are reloaded at the completion of the DMA cycle in which the address reload count is 1.<br>11: Reload function 2 is enabled.<br>The source address, destination address, and address reload count are reloaded at the completion of the DMA cycle in which the address reload count is 1. |
| 10, 9 | RLD1M1, 0 | Reload function 1 setting<br>Configures the reload function 1.<br>00: Reload function 1 is disabled.<br>01: Reload function 1 is enabled.<br>The source address and transfer count are reloaded at the completion of the DMA cycle in which the remaining transfer count is 1. (If the reload function 2 is enabled, the address reload count is also reloaded.)<br>10: Reload function 1 is enabled.<br>The destination address and transfer count are reloaded at the completion of the DMA cycle in which the remaining transfer count is 1. (If the reload function 2 is enabled, the address reload count is also reloaded.)<br>11: Reload function 1 is enabled.<br>The source address, destination address, and transfer count are reloaded at the completion of the DMA cycle in which the remaining transfer count is 1. (If the reload function 2 is enabled, the address reload count is also reloaded.) |
| 8, 7 | DACM1, 0 | Destination address count direction<br>Specifies the count direction of the destination address. |

| DACM1 | DACM0 | Direction of Count |
|---|---|---|
| 0 | 0 | Increment |
| 0 | 1 | Decrement |
| 1 | 0 | Fixed |
| 1 | 1 | Forbidden (No guarantee of operation) |

| 6, 5 | SACM1, 0 | Source address count direction<br>Specifies the count direction of the source address. |
|---|---|---|

| SACM1 | SACM0 | Direction of Count |
|---|---|---|
| 0 | 0 | Increment |
| 0 | 1 | Decrement |
| 1 | 0 | Fixed |
| 0 | 1 | Forbidden (No guarantee of operation) |

**Table 7.49 DTCTn Register Contents (3/3)**

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>4 to 2</td><td>DS[2:0]</td><td>Transfer data size<br>Specifies the transfer data size.
<table>
<tr><th>DS2</th><th>DS1</th><th>DS0</th><th>Transfer Data Size</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>8 bits</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>16 bits</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>32 bits</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>64 bits</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>128 bits</td></tr>
<tr><td colspan="3">Other than the above</td><td>Forbidden (No guarantee of operation)</td></tr>
</table></td></tr>
<tr><td>1, 0</td><td>TRM1, 0</td><td>Transfer mode<br>Specifies the DMA transfer mode.<br>00: Single transfer<br>01: Block transfer 1 (The number of transfers is specified by the transfer count.)<br>10: Block transfer 2 (The number of transfers is specified by the address reload count.)<br>11: Forbidden (No guarantee of operation)</td></tr>
</table>

**CAUTIONS**

1. **Except for the case to clear MLE bit, it is forbidden to write to those bits when the channel operation is enabled (DTE bit = 1). If you do, the correct operation is not guaranteed.**
2. **If forbidden settings are used for some of the bits, the correct operation is not guaranteed.**

### 7.10.2.5 DRSAn — DMAC Reload Source Address Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8410_H$ + $40_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSA31 | RSA30 | RSA29 | RSA28 | RSA27 | RSA26 | RSA25 | RSA24 | RSA23 | RSA22 | RSA21 | RSA20 | RSA19 | RSA18 | RSA17 | RSA16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSA15 | RSA14 | RSA13 | RSA12 | RSA11 | RSA10 | RSA9 | RSA8 | RSA7 | RSA6 | RSA5 | RSA4 | RSA3 | RSA2 | RSA1 | RSA0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.50 DRSAn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RSA[31:0] | Reload source address<br>Specifies the source address to be reloaded to the DMA source address register when the reload function 1 or reload function 2 is used. |

**CAUTION**

**DMA transfer for misaligned data is not supported. The possible lowest four bits of the address for each transfer data size is as follows. (× denotes an arbitrary one bit.) The correct operation is not guaranteed if you select other setting.**

| Data Size | RSA3 | RSA2 | RSA1 | RSA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 7.10.2.6 DRDAn — DMAC Reload Destination Address Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8414_H$ + $40_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDA31 | RDA30 | RDA29 | RDA28 | RDA27 | RDA26 | RDA25 | RDA24 | RDA23 | RDA22 | RDA21 | RDA20 | RDA19 | RDA18 | RDA17 | RDA16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDA15 | RDA14 | RDA13 | RDA12 | RDA11 | RDA10 | RDA9 | RDA8 | RDA7 | RDA6 | RDA5 | RDA4 | RDA3 | RDA2 | RDA1 | RDA0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.51 DRDAn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RDA[31:0] | Reload destination address<br>Specifies the destination address to be reloaded to the DMA destination address register when the reload function 1 or reload function 2 is used. |

**CAUTION**

**DMA transfer for misaligned data is not supported. The possible lowest four bits of the address for each transfer data size is as follows. (× denotes an arbitrary one bit.) The correct operation is not guaranteed if you select other setting.**

| Data Size | RDA3 | RDA2 | RDA1 | RDA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 7.10.2.7 DRTCn — DMAC Reload Transfer Count Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8418_H$ + $40_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RARC15 | RARC14 | RARC13 | RARC12 | RARC11 | RARC10 | RARC9 | RARC8 | RARC7 | RARC6 | RARC5 | RARC4 | RARC3 | RARC2 | RARC1 | RARC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RTRC15 | RTRC14 | RTRC13 | RTRC12 | RTRC11 | RTRC10 | RTRC9 | RTRC8 | RTRC7 | RTRC6 | RTRC5 | RTRC4 | RTRC3 | RTRC2 | RTRC1 | RTRC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.52 DRTCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | RARC[15:0] | Reload address reload count<br>Specifies the value to be loaded to the address reload count in the transfer count register at the timing of reload when the reload function 2 is used. |
| 15 to 0 | RTRC[15:0] | Reload transfer count<br>Specifies the value to be loaded to the transfer count in the transfer count register at the timing of reload when the reload function 1 (including a combination of reload function 1 and 2) is used. |

### 7.10.2.8 DTCCn — DMAC Transfer Count Compare Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 841C$_H$ + 40$_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMC15 | CMC14 | CMC13 | CMC12 | CMC11 | CMC10 | CMC9 | CMC8 | CMC7 | CMC6 | CMC5 | CMC4 | CMC3 | CMC2 | CMC1 | CMC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.53 DTCCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | CMC[15:0] | Transfer count compare<br>Configures the transfer count to be compared to the transfer count register. At the completion of the DMA cycle in which the remaining transfer count is the same as the value in this register, the transfer count match flag (DCSTn.CC) in the DMAC transfer status register is set. Furthermore, if the transfer count match interrupt enable (DTCTn.CCE) bit is 1, a transfer count match interrupt is generated.<br>If 0000$_H$ is set, comparison with the transfer count is disabled. In this case, the transfer count match flag in the DMAC transfer status register is never set, and a transfer count match interrupt is never generated. |

**CAUTION**

**It is forbidden to write to those bits when the channel operation is enabled (DTE bit = 1). If you do, the correct operation is not guaranteed.**

### 7.10.2.9 DCENn — DMAC Channel Operation Enable Setting Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8420_H$ + $40_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DTE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 7.54 DCENn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | DTE | Channel operation enable<br>Specifies whether to enable or disable the transfer operation of the channel. If the DTE bit is 1, DMA transfer starts when there is a DMA transfer request. If the MLE bit is 0, the DTE bit is cleared automatically at the completion of the DMA transfer. In addition, if 0 is written to the DTE bit during DMA transfer, the DMA transfer is suspended. If 1 is written to the DTE bit during suspension, the suspension is cleared and the DMA transfer resumes.<br>0: Channel operation is disabled/Channel suspended<br>1: Channel operation is enabled/Channel suspension cleared |

### 7.10.2.10 DCSTn — DMAC Transfer Status Register

**Access:** This register can be read in 32-bit units.

**Address:** FFFF $8424_H$ + $40_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ERWR | — | — | CY | ER | — | CC | TC | — | — | DR | SR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.55 DCSTn Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 12 | Reserved | When read, the value after reset is read. |
| 11 | ERWR | DMA Transfer Error occurring cycle<br>This bit is updated at the same time as setting of the DMA transfer error flag (ER), indicating which cycle of read or write the DMA transfer error occurs in. This bit is not updated when a new DMA transfer error occurs after the ER bit has been set. If the ER bit is cleared, this bit is also cleared to 0.<br>0: DMA transfer error occurs in read cycle.<br>1: DMA transfer error occurs in write cycle. |
| 10, 9 | Reserved | When read, the value after reset is read. |
| 8 | CY | DMA cycle execution state<br>This bit shows whether a DMA cycle is ongoing in this channel.<br>0: DMA cycle is not ongoing.<br>1: DMA cycle is ongoing.<br>After suspending a DMA channel, CY may be polled to assure that any possibly ongoing DMA cycle has finished. |
| 7 | ER | Transfer error flag<br>This bit is set when DMA transfer error is generated. If this bit is 1 and the DTCTn.ESE bit is set, a DMA cycle is not executed when a DMA transfer request is generated.<br>0: No DMA transfer error is generated<br>1: DMA transfer error is generated |
| 6 | Reserved | When read, the value after reset is read. |
| 5 | CC | Transfer count match flag<br>This bit is set at the completion of the DMA cycle in which the remaining transfer count is the same as the value set in the transfer compare register.<br>0: No compare match has occurred with the transfer count compare register.<br>1: Compare match has occurred with the transfer count compare register. |
| 4 | TC | Transfer completion flag<br>This bit is set at the completion of the last transfer and shows whether the DMA transfer is complete. If the MLE bit is 0 and this bit is 1, a DMA cycle is not executed when a DMA transfer request is generated.<br>0: DMA transfer incomplete<br>1: DMA transfer complete |
| 3, 2 | Reserved | When read, the value after reset is read. |

**Table 7.55 DCSTn Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | DR | Hardware DMA transfer request status<br>This bit shows whether there is a hardware DMA transfer request (DMARQ) from the DTFR.<br>This bit changes regardless of the value of the DTE bit when a hardware DMA transfer request from the DTFR is generated. If the software DMA transfer request is selected by the transfer request selection bit (DRS) in the DMAC transfer control register, this bit is not set even when a hardware DMA transfer request is input from the DTFR.<br>0: There is no hardware DMA transfer request<br>1: There is a hardware DMA transfer request |
| 0 | SR | Software DMA transfer request flag<br>This bit shows whether there is a software DMA transfer request (DMARQ).<br>This bit is automatically cleared at the completion of the last transfer. A user can set this bit by writing 1 to the SRS bit in the DMAC transfer status set register. In addition, a user can clear this bit by writing 1 to the SRC bit in the DMAC transfer status clear register, but if this is done, the ongoing DMA transfer is aborted and cannot be resumed.<br>0: There is no software DMA transfer request<br>1: There is a software DMA transfer request |

### 7.10.2.11 DCSTSn — DMAC Transfer Status Set Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8428_H$ + $40_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SRS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 7.56 DCSTSn Set Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SRS | Software DMA transfer request flag<br>A user can set the software DMA transfer request flag (SR) by writing 1 to this bit. 0 is always read from this bit. |

### 7.10.2.12 DCSTCn — DMAC Transfer Status Clear Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 842C$_H$ + 40$_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ERC | — | CCC | TCC | — | — | — | SRC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R | R | R | R/W |

**Table 7.57 DCSTCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 7 | ERC | Transfer error flag clear<br>The DMA transfer error flag (ER) can be cleared by writing 1 to this bit. 0 is always read from this bit. |
| 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 | CCC | Transfer count match flag clear<br>The transfer count match flag (CC) can be cleared by writing 1 to this bit. 0 is always read from this bit. |
| 4 | TCC | Transfer completion flag clear<br>The transfer completion flag (TC) can be cleared by writing 1 to this bit. 0 is always read from this bit. |
| 3 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SRC | Software DMA transfer request flag clear<br>The software DMA transfer request flag (SR) can be cleared by writing 1 to this bit. 0 is always read from this bit. |

### 7.10.2.13 DTFRn — DTFR Setting Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8430_H$ + $40_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | REQSEL6 | REQSEL5 | REQSEL4 | REQSEL3 | REQSEL2 | REQSEL1 | REQSEL0 | REQEN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.58 DTFRn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 7 to 1 | REQSEL[6:0] | Hardware DMA transfer source selection<br>Selects one out of 128 hardware DMA transfer sources as the hardware DMA transfer request.<br>000_0000: Selecting the DMACTRG[0] input<br>:<br>111_1111: Selecting the DMACTRG[127] input |
| 0 | REQEN | Hardware DMA transfer source selection enable<br>This bit enables/disables the hardware DMA transfer source selection.<br>0: Hardware DMA transfer source selection is disabled.<br>1: Hardware DMA transfer source selection is enabled.<br>If this bit is 0, even when the hardware DMA transfer source selected by the REQSEL6 to 0 bits is activated, it is not recognized as a hardware DMA transfer request, and a hardware DMA transfer request is not generated. |

### 7.10.2.14 DTFRRQn — DTFR Transfer Request Status Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $8434_H$ + $40_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF | DRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.59 DTFRRQn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. |
| 1 | OVF | DMA overflow flag<br>If this bit is set, the DMA channel has received at least one more DMA requests than it can store or handle. One or more DMA requests are lost.<br>OVF can only be read, not written. Write 1 to DTFRRQCn.OVFC to clear OVF. |
| 0 | DRQ | Hardware DMA transfer request status<br>If this bit is set, it means that a hardware DMA transfer request exists or is retained.<br>• If the hardware DMA transfer request is an edge detection type*1 This bit shows whether a hardware DMA transfer request generated by edge detection is retained. When the DMA transfer request acceptance signal (DMAAKn) from the DMAC is asserted, this bit is automatically cleared. A user can clear this bit by writing 1 to the DRQCn.DRQC bit.<br>• If the hardware DMA transfer request is a level input type*1<br>This bit shows whether there is a hardware DMA transfer request input from the outside. Even when the DMA transfer request acceptance signal (DMAAKn) from the DMAC is asserted, this bit is not automatically cleared. In addition, this bit is not cleared even when a user writes to the DRQCn.DRQC bit.<br>This bit changes regardless of the value of the DTFRn.REQEN bit when a hardware DMA transfer request from the outside is generated.<br>0: There is no hardware DMA transfer request<br>1: There is a hardware DMA transfer request |

Note 1. Whether the hardware DMA transfer request is an edge detection type or level input type depends on the hardware DMA transfer source selected by DTFRn.REQSEL. The hardware DMA transfer request is only an edge detection type in this device.

### 7.10.2.15 DTFRRQCn — DTFR Transfer Request Clear Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 8438$_H$ + 40$_H$ × Ch. No. n (n = 0 to 15)

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVFC | DRQC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 7.60 DTFRRQCn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | OVFC | DMA overflow flag clear<br>Write 1 to OVFC to clear DTFRRQn.OVF.<br>0 is always read from this bit. |
| 0 | DRQC | Hardware DMA transfer request clear<br>If the hardware DMA transfer request is an edge detection type*[1], a user can clear the DRQS.DRQ bit by writing 1 to this bit.<br>If the hardware DMA transfer request is a level input type*, the DRQS.DRQ bit cannot be cleared by writing to this bit.<br>0 is always read from this bit. |

Note 1. Whether the hardware DMA transfer request is an edge detection type or level input type depends on the hardware DMA transfer source selected by DTFRn.REQSEL. The hardware DMA transfer request is only an edge detection type in this device.

# 7.11 DTS Channel Register

## 7.11.1 Transfer information of the DTS (TI)

### 7.11.1.1 Structure of the TI

The transfer information of the DTS is called TI. One set of TI consists of 32 bits. 8 sets of TI are assigned for each channel. The 8 sets of TI is called TI-A, TI-B, TI-C, TI-D, TI- E, TI-F, TI-G, and TI-H.

**Figure 7.21** shows the structure of the TI.

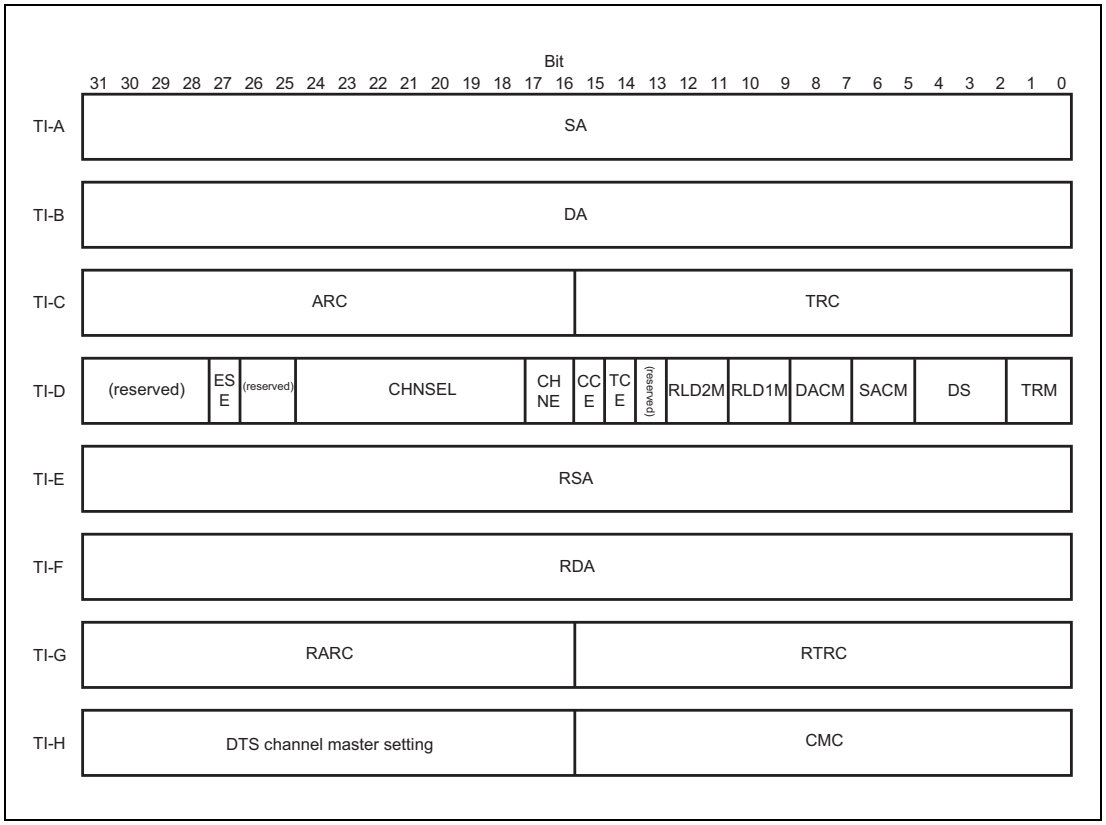



**Figure 7.21 Structure of the TI**

### 7.11.1.2 Organization of the TI in the DTSRAM

A user indirectly accesses the DTSRAM by way of the DTS channel registers for each channel and the DTS channel master setting registers.

Therefore, usually, a user does not have to think about the address organization of the TI in the DTSRAM. As an exception, when ECC error occurs while the DTSRAM is read, the address of the ECC error in the DTSRAM is stored to the DTSRAM error register 2 (DTSER2), one of the global registers. In order to know which channel and TI have generated the error, you need to understand the address organization of the TI in the DTSRAM.

**Figure 7.22** shows the address organization of the TI in the DTSRAM.

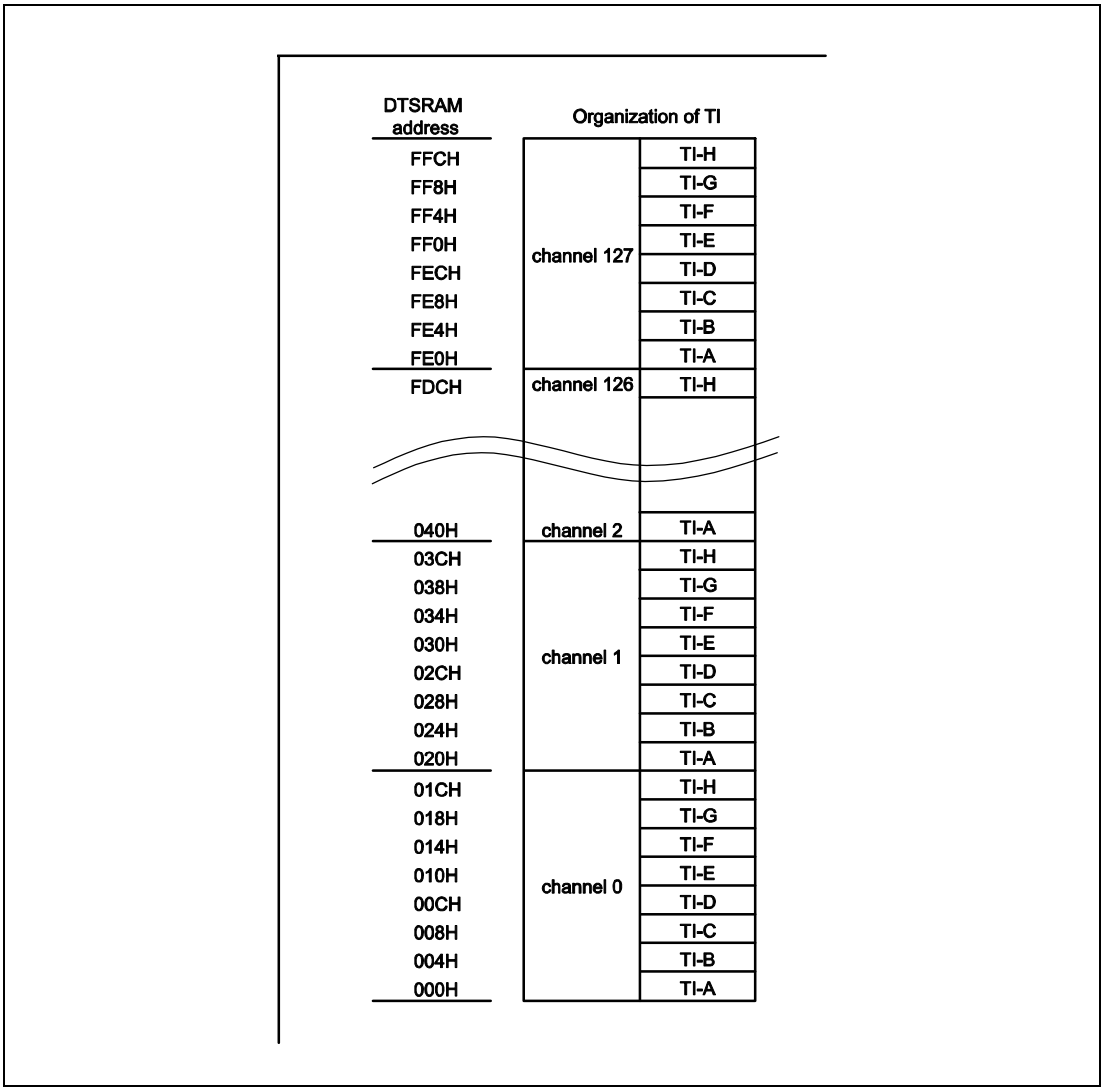



**Figure 7.22 Organization of the TI in the DTSRAM**

#### 7.11.1.3 Accessing the TI

TI-A can be accessed by way of the DTS source address register (DTSAnnn) for each channel.

TI-B can be accessed by way of the DTS destination address register (DTDAnnn) for each channel.

TI-C can be accessed by way of the DTS transfer count register (DTTCnnn) for each channel.

TI-D can be accessed by way of the DTS transfer control register (DTTCTnnn) for each channel.

TI-E can be accessed by way of the DTS reload source address register (DTRSAnnn) for each channel.

TI-F can be accessed by way of the DTS reload destination address register (DTRDAnnn) for each channel.

TI-G can be accessed by way of the DTS reload transfer count register (DTRTCnnn) for each channel.

TI-H can be accessed by way of the channel master setting register (DTSnnnCM), which is a global register, and the transfer count compare register (DTTCCnnn) for each channel.

#### 7.11.1.4 Caution about accessing the TI

The data of the DTS channel master setting register and the data of the DTS transfer count compare register are stored to the same TI-H.

Access to the DTS channel master setting register (DTSnnnCM) is actually 32-bit access to the whole TI-H. Therefore, when you write to the DTS channel master setting register, the lower 16-bit data, which is the data for the DTS transfer count compare (CMC), is updated at the same time. When you read from the DTS channel master setting register, the value of the DTS transfer count compare (CMC) is read as the lower 16-bit data.

When you read from the DTS transfer count compare register (DTTCTnnn), 32-bit data is read from the TI-H, but only the lower 16-bit data is actually seen in the result of the register read. When you write to the DTS transfer count compare register (DTTCTnnn), lower 16-bit read/modify/write access to the 32-bit TI-H is used. Data in the TI immediately after the reset is undefined. It should be noted that, if you try to write to the DTS transfer count compare register (DTTCTnnn) before setting up the DTS channel master setting register, ECC error may be detected during the read of the read/modify/write access.

The bits 28 to 23 of the TI-H are not used, but you can read and write those bits by accessing the DTS channel master setting register. Our recommendation is as follows. When you write to those bits, write 0. When you read from those bits, discard the read value by software.

You can access the TI from a CPU while the DTS is executing DMA transfer. But if you do so, the following should be noted.

- While a channel is executing DMA transfer, the TI of the channel should not be updated by TI access from a CPU. If this situation happens, the result of the DMA transfer and the contents of the TI may mismatch.
- If TI access is requested from CPU while TI fetch or TI write back is executed, the TI access is executed after the completion of TI fetch or TI write back. If TI fetch or TI write back is requested while TI access request from CPU is processed, the TI fetch or TI write back is executed after the completion of TI access.

After the reset, the values of TI in DTSRAM are undefined. After the reset, if you read TI before you write to the TI, ECC error will occur.

Therefore, the first access to the following registers after the reset must be write access. The first access after the reset should never be read access.

- DTS source address register (DTSAnnn)
- DTS destination address register (DTDAnnn)
- DTS transfer count register (DTTCnnn)
- DTS transfer control register (DTTCTnnn)
- DTS reload source address register (DTRSAnnn)
- DTS reload destination address register (DTRDAnnn)
- DTS reload transfer count register (DTRTCnnn)
- Channel master setting registers (DTSnnnCM)

In addition, the first access to DTS transfer count compare register (DTTCTnnn) after the reset must be done after write access to channel master setting register (DTSnnnCM).

You can access the TI from a CPU while the DTS is complete.executing DMA transfer. But if you do so, the following should be noted.

- While a channel is executing DMA transfer, the TI of the channel should not be updated by TI access from CPU. If this situation happens, the result of the DMA transfer and the contents of the TI may mismatch.
- If TI access is requested from CPU while TI fetch or TI write back is executed, the TI access is executed after the completion of TI fetch or TI write back. If TI fetch or TI write back is requested while TI access request from CPU is processed, the TI fetch or TI write back is executed after the completion of TI access.

### 7.11.2 DTS Channel Register Address

Address = Base address "FFFF $9000_H$" + Offset address

| Offset Address | Register Symbol | Meaning | Accessed Permission | |
|---|---|---|---|---|
| | | | Special Master | General Master |
| $0000_H + 40_H \times$ [channel number] | DTSAnnn | DTS source address | √ | √ |
| $0004_H + 40_H \times$ [channel number] | DTDAnnn | DTS destination address | √ | √ |
| $0008_H + 40_H \times$ [channel number] | DTTCnnn | DTS transfer count | √ | √ |
| $000C_H + 40_H \times$ [channel number] | DTTCTnnn | DTS transfer control | √ | √ |
| $0010_H + 40_H \times$ [channel number] | DTRSAnnn | DTS reload source address | √ | √ |
| $0014_H + 40_H \times$ [channel number] | DTRDAnnn | DTS reload destination address | √ | √ |
| $0018_H + 40_H \times$ [channel number] | DTRTCnnn | DTS reload transfer count | √ | √ |
| $001C_H + 40_H \times$ [channel number] | DTTCCnnn | DTS transfer count compare | √ | √ |
| $0020_H + 40_H \times$ [channel number] | DTFSLnnn | DTSFSL operation setting | √ | √ |
| $0024_H + 40_H \times$ [channel number] | DTFSTnnn | DTSFSL transfer request status | √ | √ |
| $0028_H + 40_H \times$ [channel number] | DTFSSnnn | DTSFSL transfer request set | √ | √ |
| $002C_H + 40_H \times$ [channel number] | DTFSCnnn | DTSFSL transfer request clear | √ | √ |

**Note:** The [channel number] in the offset addresses is a number in the range from 0 to 127.
The "nnn" in the register symbols is a 3-digit number in the range from 000 to 127.

**Note:** √: supported
x: not supported

## 7.11.3 Details of DTS Channel Registers

The “nnn” in the register symbols indicates the DTS channel number (nnn = 000 to 127).

### 7.11.3.1 DTSAnnn — DTS Source Address Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $9000_H$ + $40_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SA31 | SA30 | SA29 | SA28 | SA27 | SA26 | SA25 | SA24 | SA23 | SA22 | SA21 | SA20 | SA19 | SA18 | SA17 | SA16 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SA15 | SA14 | SA13 | SA12 | SA11 | SA10 | SA9 | SA8 | SA7 | SA6 | SA5 | SA4 | SA3 | SA2 | SA1 | SA0 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.61 DTSAnnn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SA[31:0] | Source address<br>Specifies the DMA transfer source address.<br>SA[31:0] is updated at the timing of the TI write back and retains the DMA transfer source address for the next DMA transfer. |

**CAUTION**

**DMA transfer for misaligned data is not supported. The possible lowest four bits of the address for each transfer data size is as follows. (× denotes an arbitrary one bit.) The correct operation is not guaranteed if you select other setting.**

| Data Size | SA3 | SA2 | SA1 | SA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 7.11.3.2 DTDAnnn — DTS Destination Address Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $9004_H$ + $40_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DA31 | DA30 | DA29 | DA28 | DA27 | DA26 | DA25 | DA24 | DA23 | DA22 | DA21 | DA20 | DA19 | DA18 | DA17 | DA16 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DA15 | DA14 | DA13 | DA12 | DA11 | DA10 | DA9 | DA8 | DA7 | DA6 | DA5 | DA4 | DA3 | DA2 | DA1 | DA0 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.62 DTDAnnn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DA[31:0] | Destination address<br>Specifies the DMA transfer destination address. DA[31:0] is updated at the timing of the TI write back and retains the DMA transfer destination address for the next DMA transfer. |

**CAUTIONS**

1. **If DMA transfer error is generated in the read cycle of DMA transfer, the write cycle is not executed, but the destination address is updated.**
2. **DMA transfer for misaligned data is not supported. The possible lowest four bits of the address for each transfer data size is as follows. (× denotes an arbitrary one bit.) The correct operation is not guaranteed if you select other setting.**

| Data Size | DA3 | DA2 | DA1 | DA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 7.11.3.3 DTTCnnn — DTS Transfer Count Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $9008_H$ + $40_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARC15 | ARC14 | ARC13 | ARC12 | ARC11 | ARC10 | ARC9 | ARC8 | ARC7 | ARC6 | ARC5 | ARC4 | ARC3 | ARC2 | ARC1 | ARC0 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRC15 | TRC14 | TRC13 | TRC12 | TRC11 | TRC10 | TRC9 | TRC8 | TRC7 | TRC6 | TRC5 | TRC4 | TRC3 | TRC2 | TRC1 | TRC0 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.63 DTTCnnn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | ARC[15:0] | Address reload count<br>Specifies the number of transfers until the address reload when the reload function 2 is used, and also specifies the number of transfers when the block transfer 2 is used. When the reload function 2 or block transfer 2 is used, ARC[15:0] is decremented by one for every DMA cycle and updated at the timing of the TI write back. When the reload function 2 or block transfer 2 is not used, ARC[15:0] is not updated.<br>If $0000_H$ is set, address reload is disabled. |
| 15 to 0 | TRC[15:0] | Transfer count<br>Configures the number of transfers. TRC[15:0] is decremented by one whenever a DMA cycle is executed. It is updated at the timing of the TI write back. If the reload function is not used, after the last transfer is complete, the value at the completion ($0000_H$) is retained.<br>If $0000_H$ is set, DMA transfer is not executed even when a DMA transfer request is accepted. |

| TRC15 to 0 | Operation |
|---|---|
| $0000_H$ | Transfer is disabled or complete. |
| $0001_H$ | The number of transfers or remaining transfers is 1. |
| : | : |
| $FFFF_H$ | The number of transfers or remaining transfers is 65535. |

**CAUTIONS**

1. **If transfer error is generated in the read cycle of DMA transfer, the write cycle is not executed, but the transfer count and the address reload count are updated.**
2. **Unlike a DMAC, "$0000_H$" in the transfer count of the DTS does not mean "65536 transfers" but means that transfer is disabled or complete.**

### 7.11.3.4 DTTCTnnn — DTS Transfer Control Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 900C$_H$ + 40$_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ESE | — | — | CHNSEL6 | CHNSEL5 | CHNSEL4 | CHNSEL3 | CHNSEL2 | CHNSEL1 | CHNSEL0 | CHNE1 | CHNE0 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CCE | TCE | — | RLD2M1 | RLD2M0 | RLD1M1 | RLD1M0 | DACM1 | DACM0 | SACM1 | SACM0 | DS2 | DS1 | DS0 | TRM1 | TRM0 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.64 DTTCTnnn Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | When writing, write the value “0”. When read, the unfixed value after reset and “0” after write. |
| 27 | ESE | Transfer error case DMA transfer abort setting<br>Specifies whether to abort DMA transfer when DMA transfer error is generated.<br>If this bit is cleared to 0, DMA transfer continues when DMA transfer error is generated.<br>If this bit is set to 1, DMA transfer is aborted when DMA transfer error is generated.<br>0: DMA transfer continues when DMA transfer error is generated.<br>1: DMA transfer is aborted when DMA transfer error is generated. |
| 26, 25 | Reserved | When writing, write the value “0”. When read, the unfixed value after reset and “0” after write. |
| 24 to 18 | CHNSEL[6:0] | Next channel in the chain<br>Specifies the next channel in the chain.<br>The next channel must be another channel in the DTS. You cannot specify a channel in a DMAC. You cannot specify the same channel for the next channel. (If you do, the correct operation is not guaranteed.) |
| 17, 16 | CHNE1, 0 | Chain enable<br>Selects the chain function.<br>00: Disabled<br>01: Chain at the last transfer<br>A chain request is generated at the completion of the DMA cycle in which the remaining transfer count is one.<br>10: (Forbidden. No guarantee of operation)<br>11: Always chain<br>A chain request is generated at the completion of every DMA cycle. |
| 15 | CCE | Transfer count match interrupt enable<br>If this bit is set, a transfer count match interrupt is generated at the completion of the DMA cycle in which the transfer count compare register and the remaining transfer count have the same value. |
| 14 | TCE | Transfer completion interrupt enable<br>If this bit is set, a transfer completion interrupt is generated at the completion of the last transfer. |
| 13 | Reserved | When writing, write the value “0”. When read, the unfixed value after reset and “0” after write. |

**Table 7.64 DTTCTnnn Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 12, 11 | RLD2M1, 0 | Reload function 2 setting<br>Configures the reload function 2.<br>00: Reload function 2 is disabled.<br>01: Reload function 2 is enabled.<br>The source address and address reload count are reloaded at the completion of the DMA cycle in which the address reload count is 1.<br>10: Reload function 2 is enabled.<br>The destination address and address reload count are reloaded at the completion of the DMA cycle in which the address reload count is 1.<br>11: Reload function 2 is enabled.<br>The source address, destination address, and address reload count are reloaded at the completion of the DMA cycle in which the address reload count is 1. |
| 10, 9 | RLD1M1, 0 | Reload function 1 setting<br>Configures the reload function 1.<br>00: Reload function 1 is disabled.<br>01: Reload function 1 is enabled.<br>The source address and transfer count are reloaded at the completion of the DMA cycle in which the remaining transfer count is 1. (If the reload function 2 is enabled, the address reload count is also reloaded.)<br>10: Reload function 1 is enabled.<br>The destination address and transfer count are reloaded at the completion of the DMA cycle in which the remaining transfer count is 1. (If the reload function 2 is enabled, the address reload count is also reloaded.)<br>11: Reload function 1 is enabled.<br>The source address, destination address, and transfer count are reloaded at the completion of the DMA cycle in which the remaining transfer count is 1. (If the reload function 2 is enabled, the address reload count is also reloaded.) |
| 8, 7 | DACM1, 0 | Destination address count direction<br>Specifies the count direction of the destination address. |

| DACM1 | DACM0 | Direction of Count |
|---|---|---|
| 0 | 0 | Increment |
| 0 | 1 | Decrement |
| 1 | 0 | Fixed |
| 1 | 1 | Forbidden (No guarantee of operation) |

| 6, 5 | SACM1, 0 | Source address count direction<br>Specifies the count direction of the source address. |
|---|---|---|

| SACM1 | SACM0 | Direction of Count |
|---|---|---|
| 0 | 0 | Increment |
| 0 | 1 | Decrement |
| 1 | 0 | Fixed |
| 0 | 1 | Forbidden (No guarantee of operation) |

| 4 to 2 | DS[2:0] | Transfer data size<br>Specifies the transfer data size. |
|---|---|---|

| DS2 | DS1 | DS0 | Transfer Data Size |
|---|---|---|---|
| 0 | 0 | 0 | 8 bits |
| 0 | 0 | 1 | 16 bits |
| 0 | 1 | 0 | 32 bits |
| 0 | 1 | 1 | 64 bits |
| 1 | 0 | 0 | 128 bits |
| Other than the above | | | Forbidden (No guarantee of operation) |

Table 7.64 DTTCTnnn Register Contents (3/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 1, 0 | TRM1, 0 | Transfer mode<br>Specifies the DMA transfer mode.<br>00: Single transfer<br>01: Block transfer 1 (The number of transfers is specified by the transfer count.)<br>10: Block transfer 2 (The number of transfers is specified by the address reload count.)<br>11: Forbidden (No guarantee of operation) |

**CAUTIONS**

1. **If forbidden settings are used for some of the bits, the correct operation is not guaranteed.**
2. **The 31 to 28, 26, 25, and bit 13 are unused, but you can read and write those bits. Our recommendation is as follows. When you write to those bits, write 0. When you read from those bits, discard the read value.**

### 7.11.3.5 DTRSAnnn — DTS Reload Source Address Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $9010_H$ + $40_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSA31 | RSA30 | RSA29 | RSA28 | RSA27 | RSA26 | RSA25 | RSA24 | RSA23 | RSA22 | RSA21 | RSA20 | RSA19 | RSA18 | RSA17 | RSA16 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSA15 | RSA14 | RSA13 | RSA12 | RSA11 | RSA10 | RSA9 | RSA8 | RSA7 | RSA6 | RSA5 | RSA4 | RSA3 | RSA2 | RSA1 | RSA0 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.65 DTRSAnnn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RSA[31:0] | Reload source address<br>Specifies the source address to be reloaded when the reload function 1 or reload function 2 is used. |

**CAUTION**

**DMA transfer for misaligned data is not supported. The possible lowest four bits of the address for each transfer data size is as follows. (× denotes an arbitrary one bit.) The correct operation is not guaranteed if you select other setting.**

| Data Size | RSA3 | RSA2 | RSA1 | RSA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 7.11.3.6 DTRDAnnn — DTS Reload Destination Address Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $9014_H$ + $40_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDA31 | RDA30 | RDA29 | RDA28 | RDA27 | RDA26 | RDA25 | RDA24 | RDA23 | RDA22 | RDA21 | RDA20 | RDA19 | RDA18 | RDA17 | RDA16 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDA15 | RDA14 | RDA13 | RDA12 | RDA11 | RDA10 | RDA9 | RDA8 | RDA7 | RDA6 | RDA5 | RDA4 | RDA3 | RDA2 | RDA1 | RDA0 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.66 DTRDAnnn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RDA[31:0] | Reload destination address<br>Specifies the destination address to be reloaded when the reload function 1 or reload function 2 is used. |

**CAUTION**

**DMA transfer for misaligned data is not supported. The possible lowest four bits of the address for each transfer data size is as follows. (× denotes an arbitrary one bit.) The correct operation is not guaranteed if you select other setting.**

| Data Size | RDA3 | RDA2 | RDA1 | RDA0 |
|---|---|---|---|---|
| 8 bits | × | × | × | × |
| 16 bits | × | × | × | 0 |
| 32 bits | × | × | 0 | 0 |
| 64 bits | × | 0 | 0 | 0 |
| 128 bits | 0 | 0 | 0 | 0 |

### 7.11.3.7 DTRTCnnn — DTS Reload Transfer Count Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $9018_H$ + $40_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RARC15 | RARC14 | RARC13 | RARC12 | RARC11 | RARC10 | RARC9 | RARC8 | RARC7 | RARC6 | RARC5 | RARC4 | RARC3 | RARC2 | RARC1 | RARC0 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RTRC15 | RTRC14 | RTRC13 | RTRC12 | RTRC11 | RTRC10 | RTRC9 | RTRC8 | RTRC7 | RTRC6 | RTRC5 | RTRC4 | RTRC3 | RTRC2 | RTRC1 | RTRC0 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.67 DTRTCnnn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | RARC[15:0] | Reload address reload count<br>Specifies the value to be reloaded to the address reload count when the reload function 2 is used. |
| 15 to 0 | RTRC[15:0] | Reload transfer count<br>Specifies the value to be reloaded to the transfer count when the reload function 1 or reload function 2 is used. |

| RTRC[15:0] | Operation |
|---|---|
| $0000_H$ | No DMA transfer |
| $0001_H$ | 1 transfer |
| : | : |
| $FFFF_H$ | 65535 transfers |

### 7.11.3.8 DTTCCnnn — DTS Transfer Count Compare Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $901C_H$ + $40_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMC15 | CMC14 | CMC13 | CMC12 | CMC11 | CMC10 | CMC9 | CMC8 | CMC7 | CMC6 | CMC5 | CMC4 | CMC3 | CMC2 | CMC1 | CMC0 |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.68 DTTCCnnn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | CMC[15:0] | Transfer count compare<br>Configures the transfer count to be compared to the transfer count register.<br>If the transfer count match interrupt enable (DTTCTnnn.CCE) bit in the DTS transfer control register is 1, a transfer count match interrupt is generated at the completion of the DMA cycle in which the remaining transfer count is the same as the value in this register.<br>If $0000_H$ is set, comparison with the transfer count is disabled. In this case, a transfer count match interrupt is not generated. |

**CAUTION**

**This register must be accessed after the DTS channel master setting register is set up.**

**If you access this register without setting up the DTS channel master setting register after the reset, ECC error may be generated during access to this register.**

### 7.11.3.9 DTFSLnnn — DTSFSL Operation Setting Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $9020_H$ + $40_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | REQEN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 7.69 DTFSLnnn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | REQEN | DMA transfer request enable<br>This bit selects whether a DMA transfer request from this channel retained in the DTSFSL is used as a candidate in DTS channel arbitration.<br>0: A DMA transfer request from this channel is not used as a candidate in DTS channel arbitration.<br>1: A DMA transfer request from this channel is used as a candidate in DTS channel arbitration.<br>If this bit is 0, even if the DTSFSL retains a DMA transfer request, this channel is not a candidate in DTS channel arbitration inside the DTSFSL, and consequently a DMA transfer request from this channel is not generated. |

### 7.11.3.10 DTFSTnnn — DTSFSL Transfer Request Status Register

**Access:** This register can be read in 32-bit units.

**Address:** FFFF $9024_H$ + $40_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF | DRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.70 DTFSTnnn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. |
| 1 | OVF | DTS overflow flag<br>If this bit is set, the DTS channel has received at least one more DMA requests than it can store or handle. One or more DMA requests are lost.<br>OVF can only be read, not written. Write 1 to DTFSCnnn.OVFC to clear OVF. |
| 0 | DRQ | DMA transfer request pending<br>This bit shows whether a DMA transfer request of this channel is pending.<br>This bit is set when a rising edge is detected in the hardware transfer source input from the DTSTRG pin, or when software writes “1” to the DTFSSnnn.DRQ bit.<br>This bit is automatically cleared when the DMA transfer request acceptance signal (DTSAK) from the DTS is asserted while the DTSFSL is requesting DMA transfer of this channel. Alternatively, software can clear this bit by writing 1 to the DTFSCnnn.DRQC bit.<br>0: A DMA transfer request is not pending.<br>1: A DMA transfer request is pending. |

### 7.11.3.11 DTFSSnnn — DTSFSL Transfer Request Set Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF $9028_H$ + $40_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DRQS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 7.71 DTFSSnnn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | DRQS | DMA transfer request set<br>A user can set the DTFSTnnn.DRQ bit by writing 1 to this bit.<br>0 is always read from this bit. |

### 7.11.3.12 DTFSCnnn — DTSFSL Transfer Request Clear Register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFFF 902C$_H$ + 40$_H$ × Ch. No. n (n = 0 to 127)

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVFC | DRQC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 7.72 DTFSCnnn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | OVFC | DTS overflow flag clear<br>Write 1 to OVFC to clear DTFSTnnn.OVF.<br>0 is always read from this bit. |
| 0 | DRQC | DMA transfer request clear<br>A user can clear the DTFSTnnn.DRQ bit by writing 1 to this bit.<br>0 is always read from this bit. |

## 7.12 DTS Merge Function Register

| Offset Address | Register Symbol | Meaning |
|---|---|---|
| FFD4 0000$_H$ | PINT0 | DTS interrupt status register 0 |
| FFD4 0004$_H$ | PINT1 | DTS interrupt status register 1 |
| FFD4 0008$_H$ | PINT2 | DTS interrupt status register 2 |
| FFD4 000C$_H$ | PINT3 | DTS interrupt status register 3 |
| FFD4 0010$_H$ | PINT4 | DTS interrupt status register 4 |
| FFD4 0014$_H$ | PINT5 | DTS interrupt status register 5 |
| FFD4 0018$_H$ | PINT6 | DTS interrupt status register 6 |
| FFD4 001C$_H$ | PINT7 | DTS interrupt status register 7 |
| FFD4 0020$_H$ | PINTCLR0 | DTS interrupt clear register 0 |
| FFD4 0024$_H$ | PINTCLR1 | DTS interrupt clear register 1 |
| FFD4 0028$_H$ | PINTCLR2 | DTS interrupt clear register 2 |
| FFD4 002C$_H$ | PINTCLR3 | DTS interrupt clear register 3 |
| FFD4 0030$_H$ | PINTCLR4 | DTS interrupt clear register 4 |
| FFD4 0034$_H$ | PINTCLR5 | DTS interrupt clear register 5 |
| FFD4 0038$_H$ | PINTCLR6 | DTS interrupt clear register 6 |
| FFD4 003C$_H$ | PINTCLR7 | DTS interrupt clear register 7 |

## 7.12.1 DTS Merge Function Register Descriptions

### 7.12.1.1 PINT0 — DTS Interrupt Status Register 0

The DTS transfer completion interrupt and transfer count match interrupt are merged in 32-channel units. PINT0 contain the interrupt status flags to indicate the originating channels for these interrupts.

Writing the interrupt read register value to the interrupt clear register (PINTCLR0) of the same channel in the interrupt handler clears interrupts. Interrupts on the lower-bits side take precedence to support multiplex interrupts.

**Access:** This register can be read in 32-bit units.

**Address:** FFD4 $0000_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTDTS31 | INTDTS30 | INTDTS29 | INTDTS28 | INTDTS27 | INTDTS26 | INTDTS25 | INTDTS24 | INTDTS23 | INTDTS22 | INTDTS21 | INTDTS20 | INTDTS19 | INTDTS18 | INTDTS17 | INTDTS16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTDTS15 | INTDTS14 | INTDTS13 | INTDTS12 | INTDTS11 | INTDTS10 | INTDTS9 | INTDTS8 | INTDTS7 | INTDTS6 | INTDTS5 | INTDTS4 | INTDTS3 | INTDTS2 | INTDTS1 | INTDTS0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.73 PINT0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTDTS[31:0] | Data transfer completion interrupt generated<br>INTDTSn indicates the status of a data transfer completion interrupt for DTS channel n.<br>INTDTSn = 0: No interrupt generated<br>INTDTSn = 1: Transfer completion interrupt for channel n generated<br>**NOTE**<br>If multiple transfer completion interrupts are pending at the same time, only the INTDTSn bit for the channel with the highest priority (lowest channel number) is set.<br>The PINTn registers are read only registers. INTDTSn is cleared by writing 1 to the corresponding INTCLRn bit of the PINTCLR0 register.<br>There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT0 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.2 PINT1 — DTS Interrupt Status Register 1

The DTS transfer completion interrupt and transfer count match interrupt are merged in 32-channel units. PINT1 contain the interrupt status flags to indicate the originating channels for these interrupts.

Writing the interrupt read register value to the interrupt clear register (PINTCLR1) of the same channel in the interrupt handler clears interrupts. Interrupts on the lower-bits side take precedence to support multiplex interrupts.

**Access:** This register can be read in 32-bit units.

**Address:** FFD4 $0004_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTDTS63 | INTDTS62 | INTDTS61 | INTDTS60 | INTDTS59 | INTDTS58 | INTDTS57 | INTDTS56 | INTDTS55 | INTDTS54 | INTDTS53 | INTDTS52 | INTDTS51 | INTDTS50 | INTDTS49 | INTDTS48 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTDTS47 | INTDTS46 | INTDTS45 | INTDTS44 | INTDTS43 | INTDTS42 | INTDTS41 | INTDTS40 | INTDTS39 | INTDTS38 | INTDTS37 | INTDTS36 | INTDTS35 | INTDTS34 | INTDTS33 | INTDTS32 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.74 PINT1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTDTS[63:32] | Data transfer completion interrupt generated<br>INTDTSn indicates the status of a data transfer completion interrupt for DTS channel n.<br>INTDTSn = 0: No interrupt generated<br>INTDTSn = 1: Transfer completion interrupt for channel n generated<br>**NOTE**<br>If multiple transfer completion interrupts are pending at the same time, only the INTDTSn bit for the channel with the highest priority (lowest channel number) is set.<br>The PINTn registers are read only registers. INTDTSn is cleared by writing 1 to the corresponding INTCLRn bit of the PINTCLR1 register.<br>There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT1 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.3 PINT2 — DTS Interrupt Status Register 2

The DTS transfer completion interrupt and transfer count match interrupt are merged in 32-channel units. PINT2 contain the interrupt status flags to indicate the originating channels for these interrupts.

Writing the interrupt read register value to the interrupt clear register (PINTCLR2) of the same channel in the interrupt handler clears interrupts. Interrupts on the lower-bits side take precedence to support multiplex interrupts.

**Access:** This register can be read in 32-bit units.

**Address:** FFD4 $0008_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTDTS95 | INTDTS94 | INTDTS93 | INTDTS92 | INTDTS91 | INTDTS90 | INTDTS89 | INTDTS88 | INTDTS87 | INTDTS86 | INTDTS85 | INTDTS84 | INTDTS83 | INTDTS82 | INTDTS81 | INTDTS80 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTDTS79 | INTDTS78 | INTDTS77 | INTDTS76 | INTDTS75 | INTDTS74 | INTDTS73 | INTDTS72 | INTDTS71 | INTDTS70 | INTDTS69 | INTDTS68 | INTDTS67 | INTDTS66 | INTDTS65 | INTDTS64 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.75 PINT2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTDTS[95:64] | Data transfer completion interrupt generated<br>INTDTSn indicates the status of a data transfer completion interrupt for DTS channel n.<br>INTDTSn = 0: No interrupt generated<br>INTDTSn = 1: Transfer completion interrupt for channel n generated<br>**NOTE**<br>If multiple transfer completion interrupts are pending at the same time, only the INTDTSn bit for the channel with the highest priority (lowest channel number) is set.<br>The PINTn registers are read only registers. INTDTSn is cleared by writing 1 to the corresponding INTCLRn bit of the PINTCLR2 register.<br>There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT2 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.4 PINT3 — DTS Interrupt Status Register 3

The DTS transfer completion interrupt and transfer count match interrupt are merged in 32-channel units. PINT3 contain the interrupt status flags to indicate the originating channels for these interrupts.

Writing the interrupt read register value to the interrupt clear register (PINTCLR3) of the same channel in the interrupt handler clears interrupts. Interrupts on the lower-bits side take precedence to support multiplex interrupts.

**Access:** This register can be read in 32-bit units.

**Address:** FFD4 000C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTDTS127 | INTDTS126 | INTDTS125 | INTDTS124 | INTDTS123 | INTDTS122 | INTDTS121 | INTDTS120 | INTDTS119 | INTDTS118 | INTDTS117 | INTDTS116 | INTDTS115 | INTDTS114 | INTDTS113 | INTDTS112 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTDTS111 | INTDTS110 | INTDTS109 | INTDTS108 | INTDTS107 | INTDTS106 | INTDTS105 | INTDTS104 | INTDTS103 | INTDTS102 | INTDTS101 | INTDTS100 | INTDTS99 | INTDTS98 | INTDTS97 | INTDTS96 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.76 PINT3 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTDTS[127:96] | Data transfer completion interrupt generated<br>INTDTSn indicates the status of a data transfer completion interrupt for DTS channel n.<br>INTDTSn = 0: No interrupt generated<br>INTDTSn = 1: Transfer completion interrupt for channel n generated<br>**NOTE**<br>If multiple transfer completion interrupts are pending at the same time, only the INTDTSn bit for the channel with the highest priority (lowest channel number) is set.<br>The PINTn registers are read only registers. INTDTSn is cleared by writing 1 to the corresponding INTCLRn bit of the PINTCLR3 register.<br>There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT3 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.5 PINT4 — DTS interrupt status Register 4

The DTS transfer completion interrupt and transfer count match interrupt are merged in 32-channel units. PINT4 contain the interrupt status flags to indicate the originating channels for these interrupts.

Writing the interrupt read register value to the interrupt clear register (PINTCLR4) of the same channel in the interrupt handler clears interrupts. Interrupts on the lower-bits side take precedence to support multiplex interrupts.

**Access:** This register can be read in 32-bit units.

**Address:** FFD4 0010$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCTDTS31 | INTCTDTS30 | INTCTDTS29 | INTCTDTS28 | INTCTDTS27 | INTCTDTS26 | INTCTDTS25 | INTCTDTS24 | INTCTDTS23 | INTCTDTS22 | INTCTDTS21 | INTCTDTS20 | INTCTDTS19 | INTCTDTS18 | INTCTDTS17 | INTCTDTS16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCTDTS15 | INTCTDTS14 | INTCTDTS13 | INTCTDTS12 | INTCTDTS11 | INTCTDTS10 | INTCTDTS9 | INTCTDTS8 | INTCTDTS7 | INTCTDTS6 | INTCTDTS5 | INTCTDTS4 | INTCTDTS3 | INTCTDTS2 | INTCTDTS1 | INTCTDTS0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.77 PINT4 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCTDTS[31:0] | DTS count match interrupt generated<br>INTCTDTSn indicates the status of a DTS count match interrupt for DTS channel n.<br>INTCTDTSn = 0: No interrupt generated<br>INTCTDTSn = 1: Count match interrupt for channel n generated<br>**NOTE**<br>If multiple count match interrupts are pending at the same time, only the INTCTDTSn bit for the channel with the highest priority (lowest channel number) is set.<br>The PINTn registers are read only registers. INTCTDTSn is cleared by writing 1 to the corresponding INTCLRn bit of the PINTCLR4 register.<br>There is some delay between writing the INTCLRn bit and clearing the INTCTDTSn flag. It is recommended to read PINT4 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.6 PINT5 — DTS interrupt status Register 5

The DTS transfer completion interrupt and transfer count match interrupt are merged in 32-channel units. PINT5 contain the interrupt status flags to indicate the originating channels for these interrupts.

Writing the interrupt read register value to the interrupt clear register (PINTCLR5) of the same channel in the interrupt handler clears interrupts. Interrupts on the lower-bits side take precedence to support multiplex interrupts.

**Access:** This register can be read in 32-bit units.

**Address:** FFD4 $0014_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCTDTS63 | INTCTDTS62 | INTCTDTS61 | INTCTDTS60 | INTCTDTS59 | INTCTDTS58 | INTCTDTS57 | INTCTDTS56 | INTCTDTS55 | INTCTDTS54 | INTCTDTS53 | INTCTDTS52 | INTCTDTS51 | INTCTDTS50 | INTCTDTS49 | INTCTDTS48 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCTDTS47 | INTCTDTS46 | INTCTDTS45 | INTCTDTS44 | INTCTDTS43 | INTCTDTS42 | INTCTDTS41 | INTCTDTS40 | INTCTDTS39 | INTCTDTS38 | INTCTDTS37 | INTCTDTS36 | INTCTDTS35 | INTCTDTS34 | INTCTDTS33 | INTCTDTS32 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.78 PINT5 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCTDTS [63:32] | DTS count match interrupt generated<br>INTCTDTSn indicates the status of a DTS count match interrupt for DTS channel n.<br>INTCTDTSn = 0: No interrupt generated<br>INTCTDTSn = 1: Count match interrupt for channel n generated<br>**NOTE**<br>If multiple count match interrupts are pending at the same time, only the INTCTDTSn bit for the channel with the highest priority (lowest channel number) is set.<br>The PINTn registers are read only registers. INTCTDTSn is cleared by writing 1 to the corresponding INTCLRn bit of the PINTCLR5 register.<br>There is some delay between writing the INTCLRn bit and clearing the INTCTDTSn flag. It is recommended to read PINT5 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.7 PINT6 — DTS interrupt status Register 6

The DTS transfer completion interrupt and transfer count match interrupt are merged in 32-channel units. PINT6 contain the interrupt status flags to indicate the originating channels for these interrupts.

Writing the interrupt read register value to the interrupt clear register (PINTCLR6) of the same channel in the interrupt handler clears interrupts. Interrupts on the lower-bits side take precedence to support multiplex interrupts.

**Access:** This register can be read in 32-bit units.

**Address:** FFD4 0018$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCTDTS95 | INTCTDTS94 | INTCTDTS93 | INTCTDTS92 | INTCTDTS91 | INTCTDTS90 | INTCTDTS89 | INTCTDTS88 | INTCTDTS87 | INTCTDTS86 | INTCTDTS85 | INTCTDTS84 | INTCTDTS83 | INTCTDTS82 | INTCTDTS81 | INTCTDTS80 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCTDTS79 | INTCTDTS78 | INTCTDTS77 | INTCTDTS76 | INTCTDTS75 | INTCTDTS74 | INTCTDTS73 | INTCTDTS72 | INTCTDTS71 | INTCTDTS70 | INTCTDTS69 | INTCTDTS68 | INTCTDTS67 | INTCTDTS66 | INTCTDTS65 | INTCTDTS64 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.79 PINT6 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCTDTS [95:64] | DTS count match interrupt generated<br>INTCTDTSn indicates the status of a DTS count match interrupt for DTS channel n.<br>INTCTDTSn = 0: No interrupt generated<br>INTCTDTSn = 1: Count match interrupt for channel n generated<br>**NOTE**<br>If multiple count match interrupts are pending at the same time, only the INTCTDTSn bit for the channel with the highest priority (lowest channel number) is set.<br>The PINTn registers are read only registers. INTCTDTSn is cleared by writing 1 to the corresponding INTCLRn bit of the PINTCLR6 register.<br>There is some delay between writing the INTCLRn bit and clearing the INTCTDTSn flag. It is recommended to read PINT6 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.8 PINT7 — DTS interrupt status Register 7

The DTS transfer completion interrupt and transfer count match interrupt are merged in 32-channel units. PINT7 contain the interrupt status flags to indicate the originating channels for these interrupts.

Writing the interrupt read register value to the interrupt clear register (PINTCLR7) of the same channel in the interrupt handler clears interrupts. Interrupts on the lower-bits side take precedence to support multiplex interrupts.

**Access:** This register can be read in 32-bit units.

**Address:** FFD4 001C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCTDTS127 | INTCTDTS126 | INTCTDTS125 | INTCTDTS124 | INTCTDTS123 | INTCTDTS122 | INTCTDTS121 | INTCTDTS120 | INTCTDTS119 | INTCTDTS118 | INTCTDTS117 | INTCTDTS116 | INTCTDTS115 | INTCTDTS114 | INTCTDTS113 | INTCTDTS112 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCTDTS111 | INTCTDTS110 | INTCTDTS109 | INTCTDTS108 | INTCTDTS107 | INTCTDTS106 | INTCTDTS105 | INTCTDTS104 | INTCTDTS103 | INTCTDTS102 | INTCTDTS101 | INTCTDTS100 | INTCTDTS99 | INTCTDTS98 | INTCTDTS97 | INTCTDTS96 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.80 PINT7 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCTDTS [127:96] | DTS count match interrupt generated<br>INTCTDTSn indicates the status of a DTS count match interrupt for DTS channel n.<br>INTCTDTSn = 0: No interrupt generated<br>INTCTDTSn = 1: Count match interrupt for channel n generated<br>**NOTE**<br>If multiple count match interrupts are pending at the same time, only the INTCTDTSn bit for the channel with the highest priority (lowest channel number) is set.<br>The PINTn registers are read only registers. INTCTDTSn is cleared by writing 1 to the corresponding INTCLRn bit of the PINTCLR7 register.<br>There is some delay between writing the INTCLRn bit and clearing the INTCTDTSn flag. It is recommended to read PINT7 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.9 PINTCLR0 — DTS interrupt clear Register 0

**Access:** This register can be written in 32-bit units.

**Address:** FFD4 $0020_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 31 | INTCLR 30 | INTCLR 29 | INTCLR 28 | INTCLR 27 | INTCLR 26 | INTCLR 25 | INTCLR 24 | INTCLR 23 | INTCLR 22 | INTCLR 21 | INTCLR 20 | INTCLR 19 | INTCLR 18 | INTCLR 17 | INTCLR 16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 15 | INTCLR 14 | INTCLR 13 | INTCLR 12 | INTCLR 11 | INTCLR 10 | INTCLR 9 | INTCLR 8 | INTCLR 7 | INTCLR 6 | INTCLR 5 | INTCLR 4 | INTCLR 3 | INTCLR 2 | INTCLR 1 | INTCLR 0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 7.81 PINTCLR0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCLR [31:0] | Data transfer completion interrupt clear<br>Writing 1 to the INTCLRn bit clears the corresponding INTDTSn flag of the PINT0 register.<br>INTCLRn = 0: Don't clear INTDTSn flag<br>INTCLRn = 1: Clear INTDTSn flag<br>**NOTE**<br>1. These bits are always read as 0.<br>2. There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT0 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.10 PINTCLR1 — DTS interrupt clear Register 1

**Access:** This register can be read/written in 32-bit units.

**Address:** FFD4 $0024_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 63 | INTCLR 62 | INTCLR 61 | INTCLR 60 | INTCLR 59 | INTCLR 58 | INTCLR 57 | INTCLR 56 | INTCLR 55 | INTCLR 54 | INTCLR 53 | INTCLR 52 | INTCLR 51 | INTCLR 50 | INTCLR 49 | INTCLR 48 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 47 | INTCLR 46 | INTCLR 45 | INTCLR 44 | INTCLR 43 | INTCLR 42 | INTCLR 41 | INTCLR 40 | INTCLR 39 | INTCLR 38 | INTCLR 37 | INTCLR 36 | INTCLR 35 | INTCLR 34 | INTCLR 33 | INTCLR 32 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.82 PINTCLR1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCLR [63:32] | Data transfer completion interrupt clear<br>Writing 1 to the INTCLRn bit clears the corresponding INTDTSn flag of the PINT1 register.<br>INTCLRn = 0: Don't clear INTDTSn flag<br>INTCLRn = 1: Clear INTDTSn flag<br>**NOTE**<br>1. These bits are always read as 0.<br>2. There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT1 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.11 PINTCLR2 — DTS interrupt clear Register 2

**Access:** This register can be read/written in 32-bit units.

**Address:** FFD4 $0028_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 95 | INTCLR 94 | INTCLR 93 | INTCLR 92 | INTCLR 91 | INTCLR 90 | INTCLR 89 | INTCLR 88 | INTCLR 87 | INTCLR 86 | INTCLR 85 | INTCLR 84 | INTCLR 83 | INTCLR 82 | INTCLR 81 | INTCLR 80 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 79 | INTCLR 78 | INTCLR 77 | INTCLR 76 | INTCLR 75 | INTCLR 74 | INTCLR 73 | INTCLR 72 | INTCLR 71 | INTCLR 70 | INTCLR 69 | INTCLR 68 | INTCLR 67 | INTCLR 66 | INTCLR 65 | INTCLR 64 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.83 PINTCLR2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCLR [95:64] | Data transfer completion interrupt clear<br>Writing 1 to the INTCLRn bit clears the corresponding INTDTSn flag of the PINT2 register.<br>INTCLRn = 0: Don't clear INTDTSn flag<br>INTCLRn = 1: Clear INTDTSn flag<br>**NOTE**<br>1. These bits are always read as 0.<br>2. There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT2 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.12 PINTCLR3 — DTS interrupt clear Register 3

**Access:** This register can be read/written in 32-bit units.

**Address:** FFD4 002C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR127 | INTCLR126 | INTCLR125 | INTCLR124 | INTCLR123 | INTCLR122 | INTCLR121 | INTCLR120 | INTCLR119 | INTCLR118 | INTCLR117 | INTCLR116 | INTCLR115 | INTCLR114 | INTCLR113 | INTCLR112 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR111 | INTCLR110 | INTCLR109 | INTCLR108 | INTCLR107 | INTCLR106 | INTCLR105 | INTCLR104 | INTCLR103 | INTCLR102 | INTCLR101 | INTCLR100 | INTCLR99 | INTCLR98 | INTCLR97 | INTCLR96 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.84 PINTCLR3 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCLR[127:96] | Data transfer completion interrupt clear<br>Writing 1 to the INTCLRn bit clears the corresponding INTDTSn flag of the PINT3 register.<br>INTCLRn = 0: Don't clear INTDTSn flag<br>INTCLRn = 1: Clear INTDTSn flag<br>**NOTE**<br>1. These bits are always read as 0.<br>2. There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT3 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.13 PINTCLR4 — DTS interrupt clear Register 4

**Access:** This register can be read/written in 32-bit units.

**Address:** FFD4 $0030_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 31 | INTCLR 30 | INTCLR 29 | INTCLR 28 | INTCLR 27 | INTCLR 26 | INTCLR 25 | INTCLR 24 | INTCLR 23 | INTCLR 22 | INTCLR 21 | INTCLR 20 | INTCLR 19 | INTCLR 18 | INTCLR 17 | INTCLR 16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 15 | INTCLR 14 | INTCLR 13 | INTCLR 12 | INTCLR 11 | INTCLR 10 | INTCLR 9 | INTCLR 8 | INTCLR 7 | INTCLR 6 | INTCLR 5 | INTCLR 4 | INTCLR 3 | INTCLR 2 | INTCLR 1 | INTCLR 0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.85 PINTCLR4 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCLR [31:0] | DTS count match interrupt clear<br>Writing 1 to the INTCLRn bit clears the corresponding INTCTDTSn flag of the PINT4 register.<br>INTCLRn = 0: Don't clear INTCTDTSn flag<br>INTCLRn = 1: Clear INTCTDTSn flag<br>**NOTE**<br>1. These bits are always read as 0.<br>2. There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT4 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.14 PINTCLR5 — DTS interrupt clear Register 5

**Access:** This register can be read/written in 32-bit units.

**Address:** FFD4 $0034_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 63 | INTCLR 62 | INTCLR 61 | INTCLR 60 | INTCLR 59 | INTCLR 58 | INTCLR 57 | INTCLR 56 | INTCLR 55 | INTCLR 54 | INTCLR 53 | INTCLR 52 | INTCLR 51 | INTCLR 50 | INTCLR 49 | INTCLR 48 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 47 | INTCLR 46 | INTCLR 45 | INTCLR 44 | INTCLR 43 | INTCLR 42 | INTCLR 41 | INTCLR 40 | INTCLR 39 | INTCLR 38 | INTCLR 37 | INTCLR 36 | INTCLR 35 | INTCLR 34 | INTCLR 33 | INTCLR 32 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.86 PINTCLR5 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCLR [63:32] | DTS count match interrupt clear<br>Writing 1 to the INTCLRn bit clears the corresponding INTCTDTSn flag of the PINT5 register.<br>INTCLRn = 0: Don't clear INTCTDTSn flag<br>INTCLRn = 1: Clear INTCTDTSn flag<br>**NOTE**<br>1. These bits are always read as 0.<br>2. There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT5 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.15 PINTCLR6 — DTS interrupt clear Register 6

**Access:** This register can be read/written in 32-bit units.

**Address:** FFD4 $0038_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 95 | INTCLR 94 | INTCLR 93 | INTCLR 92 | INTCLR 91 | INTCLR 90 | INTCLR 89 | INTCLR 88 | INTCLR 87 | INTCLR 86 | INTCLR 85 | INTCLR 84 | INTCLR 83 | INTCLR 82 | INTCLR 81 | INTCLR 80 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR 79 | INTCLR 78 | INTCLR 77 | INTCLR 76 | INTCLR 75 | INTCLR 74 | INTCLR 73 | INTCLR 72 | INTCLR 71 | INTCLR 70 | INTCLR 69 | INTCLR 68 | INTCLR 67 | INTCLR 66 | INTCLR 65 | INTCLR 64 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.87 PINTCLR4 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCLR [95:64] | DTS count match interrupt clear<br>Writing 1 to the INTCLRn bit clears the corresponding INTCTDTSn flag of the PINT6 register.<br>INTCLRn = 0: Don't clear INTCTDTSn flag<br>INTCLRn = 1: Clear INTCTDTSn flag<br>**NOTE**<br>1. These bits are always read as 0.<br>2. There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT6 twice after clearing one of its bits. The second read operation will read the correct status. |

### 7.12.1.16 PINTCLR7 — DTS interrupt clear Register 7

**Access:** This register can be read/written in 32-bit units.

**Address:** FFD4 003C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR127 | INTCLR126 | INTCLR125 | INTCLR124 | INTCLR123 | INTCLR122 | INTCLR121 | INTCLR120 | INTCLR119 | INTCLR118 | INTCLR117 | INTCLR116 | INTCLR115 | INTCLR114 | INTCLR113 | INTCLR112 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLR111 | INTCLR110 | INTCLR109 | INTCLR108 | INTCLR107 | INTCLR106 | INTCLR105 | INTCLR104 | INTCLR103 | INTCLR102 | INTCLR101 | INTCLR100 | INTCLR99 | INTCLR98 | INTCLR97 | INTCLR96 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.88 PINTCLR7 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | INTCLR[127:96] | DTS count match interrupt clear<br>Writing 1 to the INTCLRn bit clears the corresponding INTCTDTSn flag of the PINT7 register.<br>INTCLRn = 0: Don't clear INTCTDTSn flag<br>INTCLRn = 1: Clear INTCTDTSn flag<br>**NOTE**<br>1. These bits are always read as 0.<br>2. There is some delay between writing the INTCLRn bit and clearing the INTDTSn flag. It is recommended to read PINT7 twice after clearing one of its bits. The second read operation will read the correct status. |

# 7.13 DTS Trigger Select Registers

| Offset Address | Register Symbol | Meaning |
|---|---|---|
| FFD4 $1000_H$ | DTSTRGSEL0 | DTS primary/secondary select register 0 |
| FFD4 $1004_H$ | DTSTRGSEL1 | DTS primary/secondary select register 1 |
| FFD4 $1008_H$ | DTSTRGSEL2 | DTS primary/secondary select register 2 |
| FFD4 $100C_H$ | DTSTRGSEL3 | DTS primary/secondary select register 3 |

## 7.13.1 DTS Trigger Select Register Descriptions

### 7.13.1.1 DTSTRGSEL0 — DTS primary/secondary select register 0

**Access:** This register can be read/written in 32-bit units.

**Address:** FFD4 $1000_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTSSEL31 | DTSSEL30 | DTSSEL29 | DTSSEL28 | DTSSEL27 | DTSSEL26 | DTSSEL25 | DTSSEL24 | DTSSEL23 | DTSSEL22 | DTSSEL21 | DTSSEL20 | DTSSEL19 | DTSSEL18 | DTSSEL17 | DTSSEL16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTSSEL15 | DTSSEL14 | DTSSEL13 | DTSSEL12 | DTSSEL11 | DTSSEL10 | DTSSEL9 | DTSSEL8 | DTSSEL7 | DTSSEL6 | DTSSEL5 | DTSSEL4 | DTSSEL3 | DTSSEL2 | DTSSEL1 | DTSSEL0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.89 DTSTRGSEL0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTSSEL[31:0] | Trigger selector<br>DTSSELn specifies the DTS trigger input for DTS channel n.<br>DTSSELn = 0: Primary channel selected<br>DTSSELn = 1: Secondary channel selected. If a secondary channel is not available, the DTS trigger is kept at low level (inactive). |

### 7.13.1.2 DTSTRGSEL1 — DTS primary/secondary select register 1

**Access:** This register can be read/written in 32-bit units.

**Address:** FFD4 $1004_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTSSEL63 | DTSSEL62 | DTSSEL61 | DTSSEL60 | DTSSEL59 | DTSSEL58 | DTSSEL57 | DTSSEL56 | DTSSEL55 | DTSSEL54 | DTSSEL53 | DTSSEL52 | DTSSEL51 | DTSSEL50 | DTSSEL49 | DTSSEL48 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTSSEL47 | DTSSEL46 | DTSSEL45 | DTSSEL44 | DTSSEL43 | DTSSEL42 | DTSSEL41 | DTSSEL40 | DTSSEL39 | DTSSEL38 | DTSSEL37 | DTSSEL36 | DTSSEL35 | DTSSEL34 | DTSSEL33 | DTSSEL32 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.90 DTSTRGSEL1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTSSEL[63:32] | Trigger selector<br>DTSSELn specifies the DTS trigger input for DTS channel n.<br>DTSSELn = 0: Primary channel selected<br>DTSSELn = 1: Secondary channel selected. If a secondary channel is not available, the DTS trigger is kept at low level (inactive). |

### 7.13.1.3 DTSTRGSEL2 — DTS primary/secondary select register 2

**Access:** This register can be read/written in 32-bit units.

**Address:** FFD4 $1008_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTSSEL95 | DTSSEL94 | DTSSEL93 | DTSSEL92 | DTSSEL91 | DTSSEL90 | DTSSEL89 | DTSSEL88 | DTSSEL87 | DTSSEL86 | DTSSEL85 | DTSSEL84 | DTSSEL83 | DTSSEL82 | DTSSEL81 | DTSSEL80 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTSSEL79 | DTSSEL78 | DTSSEL77 | DTSSEL76 | DTSSEL75 | DTSSEL74 | DTSSEL73 | DTSSEL72 | DTSSEL71 | DTSSEL70 | DTSSEL69 | DTSSEL68 | DTSSEL67 | DTSSEL66 | DTSSEL65 | DTSSEL64 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.91 DTSTRGSEL2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTSSEL[95:64] | Trigger selector<br>DTSSELn specifies the DTS trigger input for DTS channel n.<br>DTSSELn = 0: Primary channel selected<br>DTSSELn = 1: Secondary channel selected. If a secondary channel is not available, the DTS trigger is kept at low level (inactive). |

### 7.13.1.4 DTSTRGSEL3 — DTS primary/secondary select register 3

**Access:** This register can be read/written in 32-bit units.

**Address:** FFD4 100C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTSSEL127 | DTSSEL126 | DTSSEL125 | DTSSEL124 | DTSSEL123 | DTSSEL122 | DTSSEL121 | DTSSEL120 | DTSSEL119 | DTSSEL118 | DTSSEL117 | DTSSEL116 | DTSSEL115 | DTSSEL114 | DTSSEL113 | DTSSEL112 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTSSEL111 | DTSSEL110 | DTSSEL109 | DTSSEL108 | DTSSEL107 | DTSSEL106 | DTSSEL105 | DTSSEL104 | DTSSEL103 | DTSSEL102 | DTSSEL101 | DTSSEL100 | DTSSEL99 | DTSSEL98 | DTSSEL97 | DTSSEL96 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 7.92 DTSTRGSEL3 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DTSSEL[127:96] | Trigger selector<br>DTSSELn specifies the DTS trigger input for DTS channel n.<br>DTSSELn = 0: Primary channel selected<br>DTSSELn = 1: Secondary channel selected. If a secondary channel is not available, the DTS trigger is kept at low level (inactive). |

## 7.14 H-Bus DMA Halt Function

H-Bus Peripheral Master has DMA function. H-Bus DMA Halt function can halt transfer to/from memories (Global RAM, PE1/2 Local RAM, Code Flash) of H-Bus Peripheral Master.

A halt can be requested via the HTHDMACTL register. Transfer to Global RAM and to the other memories (PE1/2 Local RAM, Code Flash) can be halted individually. To check if all transfers have been halted, HTHDMASTAT register can be used.

### 7.14.1 H-Bus DMA Halt register Address

| Address | Register Symbol | Meaning |
|---|---|---|
| FFFA 2000$_H$ | HTHDMACTL | H-Bus DMA Halt control register |
| FFFA 2004$_H$ | HTHDMASTAT | H-Bus DMA Halt status register |

### 7.14.2 Detail of H-Bus DMA Halt register

#### 7.14.2.1 HTHDMACTL — H-Bus DMA Halt Control register

**Access:** This register can be read/written in 8/16/32-bit units.

**Address:** FFFA 2000$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SICSRQ | GRISRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 7.93 HTHDMACTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SICSRQ | Halt request for H-Bus DMA transfer to/from LRAM and Code Flash<br>0: System interconnect target is in normal operation mode<br>1: System interconnect target is in halt mode |
| 0 | GRISRQ | Halt request for H-Bus DMA transfer to/from GRAM<br>0: GRAM target is in normal operation mode<br>1: GRAM target is in halt mode |

### 7.14.2.2 HTHDMASTAT — H-Bus DMA Halt status register

**Access:** This register can be read/written in 8/16/32-bit units.

**Address:** FFFA $2004_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SICSAK | GRISAK |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 7.94 HTHDMASTAT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SICSAK | Halt status for H-Bus DMA transfer to/from LRAM and Code Flash<br>0: System interconnect target has continued operation<br>1: System interconnect target has stopped operation |
| 0 | GRISAK | Halt status for H-Bus DMA transfer to/from GRAM<br>0: GRAM target has continued operation<br>1: GRAM target has stopped operation |

**CAUTION**

Please do not clear HTHDMACTL halt request bit until HTHDMASTAT halt status bit will be set, once you sets HTHDMACTL halt request bit. Also, please do not set HTHDMACTL halt request bit until HTHDMASTAT halt status bit will be clear, once you clears HTHDMACTL halt request bit.

Please note that all H-Bus Peripheral Master have built-in memories to buffer received messages while the bus is stopped. The user is responsible that these buffers do not overflow while the H-Bus is halted.

## 7.15 Difference among P1M-C, P1H-C and P1H-CE

There are some differences of available DMA/DTS trigger sources listed in **Section 7.7.1** and **Section 7.8.1**.

# Section 8 Reset Controller

## 8.1 Features

The Reset Controller controls all factors that have an influence on the reset behavior of the device.

The device has several kinds of reset categories depending on the areas that are reset. Each of reset categories is triggered by one or multiple reset sources.

The relation between reset categories and their sources are shown in **Table 8.1**. The relation between reset categories and initialized area is shown in **Table 8.2**.

**Table 8.1 Reset Categories and Reset Sources**

| Reset Category | Source |
|---|---|
| Power On Reset | • Power On Reset*1 |
| System Reset 1 | • Terminal Reset<br>• CVM Reset |
| System Reset 2 | • Software Reset (by SWSRESA0, SWSRESA1)<br>• ECM Reset (if RESC0 = 0) |
| Application Reset 1 | • Software Reset (by SWARESA0, SWARESA1)<br>• ECM Reset (if RESC0 = 1) |
| Limited Reset | • Software Reset (by SWLRESAn)<br>• ICUMC WDTA Reset |
| Debug Reset | • Debug Reset (by TRSTZ)*2 |

Note 1. Power On Reset is asserted when SYSVCC supply voltage is under VPOC level and is released when SYSVCC supply is over VPOC level.

Note 2. For details, See **Section 31, On-Chip Debugging Unit (OCD)**.

**Table 8.2 Reset categories and initialized area ( √ : reset (initialized) — : without reset (kept) )**

| Initialized Area(module) | | Reset category | | | | |
|---|---|---|---|---|---|---|
| | | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| PE1 | PE1 | √ | √ | √ | √ | — |
| | PE1 Local RAM (backup section) | √ | √ | √ | √*1 | — |
| | PE1 Local RAM (excluding backup section) | √ | √ | √ | √*1 | — |
| | Window Watchdog Timer 0 | √ | √ | √ | √ | — |
| PE2 | PE2 | √ | √ | √ | √ | — |
| | PE2 Local RAM | √ | √ | √ | √*1 | — |
| | Window Watchdog Timer 1 | √ | √ | √ | √ | — |
| ICUMC | ICUMC | √ | √ | √ | √ | √ |
| | ICUMC Local RAM | √ | √ | √ | √*1 | √*1 |
| | ICUMC Local Watchdog Timer (WDTA) | √ | √ | √ | √ | √ |
| | ICUMC Debug Logic | √ | √ | √ | √ | — |
| RAM | Global RAM | √ | √ | √ | √*1 | — |
| | DTS RAM | √ | √ | √ | √*1 | — |
| System Control | Operating Mode Controller | — | √*5 | — | — | — |
| | Clock Controller (Divider0/1, Selector0) | √ | √ | — | — | — |
| | Clock Controller (Excluding Divider0/1, Selector0) | √ | √ | √ | √ | — |
| | Core Voltage Monitor | √ | √*6 | — | — | — |
| | Power Down Mode Controller | √ | √ | √ | √ | √*4 |
| | Clock Monitor | √ | √ | √ | √ | — |
| IO | IO Buffer / IO Port Function | √ | √ | √ | √ | — |
| | ERROROUTZ Pin | √*3 | √*3 | √*3 | — | — |
| | RESETOUTZ Pin | √ | √ | √ | √ | — |
| | CVMOUTZ Pin | √ | — | — | — | — |
| Peripheral | ECM master/checker error source status register 0/1/2 | √ | — | — | — | — |
| | ECM Error Output Clear Invalidation Configuration Register | √ | √ | √ | — | — |
| | ECM | √ | √ | √ | √ | — |
| | MCAN/M-TTCAN/FlexRay/GTM/ Ethernet/CSIH | √ | √ | √ | √*1 | √*1 |
| | SENT/HS-USRT/RLIN3/ADCF | √ | √ | √ | √ | √ |
| | Secure WDT (SWDT) | √ | √ | √ | √ | — |
| | System Timer (counter) | √ | √ | √ | √*2 | — |
| | System Timer (excluding counter) | √ | √ | √ | — | — |
| | Other Peripherals | √ | √ | √ | √ | — |

Note 1. The execution of RAM initialization is configurable by a register setting.
Note 2. It can be masked by a System Timer register.
Note 3. For details, see the “**Section 29, Error Control Module (ECM)**”.
Note 4. Only the associated module is reset by the individual limited reset.

Note 5. Terminal Reset only

Note 6. For details, see the “**Section 10, Core Voltage Monitor (CVM)**”.

## 8.2 Input/Output Pins

I/O pins related to reset are shown in **Table 8.3**.

**Table 8.3 I/O pins**

| Pin function name | Direction | Description |
|---|---|---|
| RESETZ | Input | Reset Input |
| RESETOUTZ | Output | Reset Output |

## 8.3 Register Description

The register list related to reset is shown in **Table 8.4**.

Registers can be protected from inadvertent write access due to erroneouts program execution, etc. by configuration of the P-Bus Guards. For details, see **Section 28, Functional Safety**.

**Table 8.4 List of Registers (1/2)**

| Address | Register Name | Description | Access Width | Value after reset | PBG |
|---|---|---|---|---|---|
| FFF8 1000h | RESF | Reset Factor Register | 32 | 0000 0403h*1<br>0000 0406h or<br>0000 0404h*2 | PBG4#0.PG4-SC3 |
| FFF8 1008h | RESFC | Reset Factor Clear Register | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 6300h | ICUMRESF | ICUMC Reset Factor Register | 32 | 0000 0000h | PBG4#2.SC1 |
| FFF8 6308h | ICUMRESFC | ICUMC Reset Factor Clear Register | 32 | 0000 0000h | PBG4#2.SC1 |
| FFF8 1100h | SWSRESA0 | Software System Reset Assertion Register 0 | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 6000h | SWSRESA1 | Software System Reset Assertion Register 1 | 32 | 0000 0000h | PBG4#2.SC4 |
| FFF8 1200h | SWARESA0 | Software Application Reset Assertion Register 0 | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 6100h | SWARESA1 | Software Application Reset Assertion Register 1 | 32 | 0000 0000h | PBG4#2.SC4 |
| FFF8 6208h | SWLRESA2 | Software Limited Reset Assertion Register for ICUMC | 32 | 0000 0000h | PBG4#2.SC1 |
| FFF8 1708h | SWLRESA3 | Software Limited Reset Assertion Register for MCAN | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1808h | SWLRESA4 | Software Limited Reset Assertion Register for FlexRay | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1908h | SWLRESA5 | Software Limited Reset Assertion Register for GTM | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1A08h | SWLRESA6 | Software Limited Reset Assertion Register for Ethernet | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1B08h | SWLRESA7 | Software Limited Reset Assertion Register for SENT | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1C08h | SWLRESA8 | Software Limited Reset Assertion Register for HS-USRT | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1E08h | SWLRESA10 | Software Limited Reset Assertion Register for CSIH | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1F08h | SWLRESA11 | Software Limited Reset Assertion Register for RLIN | 32 | 0000 0000h | PBG4#0.PG4-SC3 |

Table 8.4 List of Registers (2/2)

| Address | Register Name | Description | Access Width | Value after reset | PBG |
|---|---|---|---|---|---|
| FFF8 2008h | SWLRESA12 | Software Limited Reset Assertion Register for AD | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 620Ch | SWLRESS2 | Software Limited Reset Status Register for ICUMC | 32 | 0000 0000h | PBG4#2.SC1 |
| FFF8 170Ch | SWLRESS3 | Software Limited Reset Status Register for MCAN | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 180Ch | SWLRESS4 | Software Limited Reset Status Register for FlexRay | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 190Ch | SWLRESS5 | Software Limited Reset Status Register for GTM | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1A0Ch | SWLRESS6 | Software Limited Reset Status Register for Ethernet | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1B0Ch | SWLRESS7 | Software Limited Reset Status Register for SENT | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1C0Ch | SWLRESS8 | Software Limited Reset Status Register for HS-USRT | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1E0Ch | SWLRESS10 | Software Limited Reset Status Register for CSIH | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 1F0Ch | SWLRESS11 | Software Limited Reset Status Register for RLIN | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 200Ch | SWLRESS12 | Software Limited Reset Status Register for AD | 32 | 0000 0000h | PBG4#0.PG4-SC3 |
| FFF8 2800h | RESC | Reset Configuration Register | 32 | 0000 0001h | PBG4#0.PG4-SC3 |
| FFC5 8000h | BOOTCTRL | Boot Control Register | 32 | 0000 0000h | APBGRD_PFSS1.SP4 |
| FFF8 1320h | STAC_DTSRAM | RAM Initialization Mode Control Register for DTS-RAM | 32 | 0000 0003h | PBG4#0.PG4-SC3 |
| FFF8 1420h | STAC_GRAM | RAM Initialization Mode Control Register for GRAM | 32 | 0000 0003h | PBG4#0.PG4-SC3 |
| FFF8 1520h | STAC_LM0 | RAM Initialization Mode Control Register for PE1 | 32 | 0000 0003h | PBG4#0.PG4-SC3 |
| FFF8 1620h | STAC_LM1 | RAM Initialization Mode Control Register for PE2 | 32 | 0000 0003h | PBG4#0.PG4-SC3 |
| FFF8 6220h | STAC_LM2 | RAM Initialization Mode Control Register for ICUMC | 32 | 0000 0003h | PBG4#2.SC1 |
| FFF8 1720h | STAC_LM3 | RAM Initialization Mode Control Register for MCAN | 32 | 0000 0003h | PBG4#0.PG4-SC3 |
| FFF8 1820h | STAC_LM4 | RAM Initialization Mode Control Register for FlexRay | 32 | 0000 0003h | PBG4#0.PG4-SC3 |
| FFF8 1920h | STAC_LM5 | RAM Initialization Mode Control Register for GTM | 32 | 0000 0003h | PBG4#0.PG4-SC3 |
| FFF8 1E20h | STAC_LM10 | RAM Initialization Mode Control Register for CSIH | 32 | 0000 0003h | PBG4#0.PG4-SC3 |

Note 1. Value after Power On Reset

Note 2. Value after CVM Reset. Value of Bit 1 after CVM Reset is undefines.

Register reset condition is shown in **Table 8.5**.

**Table 8.5 Register reset condition**

| Register Name | Reset Condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| RESF | √ | √*2 | | | |
| RESFC | √ | √ | √ | √ | |
| ICUMRESF | √ | √*1 | | | |
| ICUMRESFC | √ | √ | √ | √ | |
| SWSRESA0 | √ | √ | √ | | |
| SWSRESA1 | √ | √ | √ | | |
| SWARESA0 | √ | √ | √ | | |
| SWARESA1 | √ | √ | √ | | |
| SWLRESA2 | √ | √ | √ | | |
| SWLRESA3 | √ | √ | √ | | |
| SWLRESA4 | √ | √ | √ | | |
| SWLRESA5 | √ | √ | √ | | |
| SWLRESA6 | √ | √ | √ | | |
| SWLRESA7 | √ | √ | √ | | |
| SWLRESA8 | √ | √ | √ | | |
| SWLRESA10 | √ | √ | √ | | |
| SWLRESA11 | √ | √ | √ | | |
| SWLRESA12 | √ | √ | √ | | |
| SWLRESS2 | √ | √ | √ | | |
| SWLRESS3 | √ | √ | √ | | |
| SWLRESS4 | √ | √ | √ | | |
| SWLRESS5 | √ | √ | √ | | |
| SWLRESS6 | √ | √ | √ | | |
| SWLRESS7 | √ | √ | √ | | |
| SWLRESS8 | √ | √ | √ | | |
| SWLRESS10 | √ | √ | √ | | |
| SWLRESS11 | √ | √ | √ | | |
| SWLRESS12 | √ | √ | √ | | |
| RESC | √ | √ | | | |
| BOOTCTRL | √ | √ | √ | √ | |
| STAC_DTSRAM | √ | √ | √ | | |
| STAC_GRAM | √ | √ | √ | | |
| STAC_LM0 | √ | √ | √ | | |
| STAC_LM1 | √ | √ | √ | | |
| STAC_LM2 | √ | √ | √ | | |
| STAC_LM3 | √ | √ | √ | | |
| STAC_LM4 | √ | √ | √ | | |
| STAC_LM5 | √ | √ | √ | | |
| STAC_LM10 | √ | √ | √ | | |

Note 1. CVM Reset only

Note 2. CVM Reset only: Please note the different reset value options for CVM Reset condition as described in **Table 8.4**.

### 8.3.1 RESF — Reset Factor Register

This register indicates whether a Reset occurred since it was cleared last time.

All bits will be cleared by a Power On Reset, CVM Reset or by software, except it is described differently in **Table 8.6**.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 $1000_H$

**Value after reset:** 0000 $0403_H$ : Power On Reset
0000 $0406_H$ or 0000 $0404_H$ : CVM Reset*2

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset *1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | ARESF3 | ARESF2 | ARESF1 | ARESF0 | — | SRESF4 | SRESF3 | SRESF2 | SRESF1 | SRESF0 | PRESF0 |
| Value after reset *1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.6 Reset Factor Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is read. |
| 10 | ARESF3 | HW BIST Execution Flag<br>This flag will always be asserted after a preceding Power-On or System Reset if the HW BIST is enabled. In this case the corresponding Power-On or System Reset Flag indication is asserted as well. This flag is only cleared by software.<br>0: HW BIST was not executed<br>1: HW BIST was executed |
| 9 | ARESF2 | ECM Application Reset Flag<br>0: No reset occurred<br>1: Reset has occurred |
| 8 | ARESF1 | Software Application Reset 1 Flag<br>0: No reset occurred<br>1: Reset has occurred |
| 7 | ARESF0 | Software Application Reset 0 Flag<br>0: No reset occurred<br>1: Reset has occurred |
| 6 | Reserved | When read, the value is undefined. |
| 5 | SRESF4 | ECM System Reset Flag<br>0: No reset occurred<br>1: Reset has occurred |
| 4 | SRESF3 | Software System Reset 1 Flag<br>0: No reset occurred<br>1: Reset has occurred |
| 3 | SRESF2 | Software System Reset 0 Flag<br>0: No reset occurred<br>1: Reset has occurred |
| 2 | SRESF1 | CVM Reset Flag<br>This flag is only cleared by a Power On Reset or by software.<br>0: No reset occurred<br>1: Reset has occurred |

**Table 8.6 Reset Factor Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | SRESF0*2 | Terminal Reset Flag<br>The value of this flag after CVM Reset is undefined.<br>This flag is only cleared by software.<br>0: No reset occurred<br>1: Reset has occurred |
| 0 | PRESF0 | Power On Reset Flag<br>This flag is set by a Power On Reset.<br>This flag is also set, if it is triggered by the debugger.<br>This flag can only be cleared by a CVM Reset or by software.<br>0: No reset occurred<br>1: Reset has occurred |

Note 1. Value after Power On Reset

Note 2. Value of SRESF0 after CVM Reset is undefined

### 8.3.2 RESFC — Reset Factor Clear Register

This register clears the reset flags of the RESF register.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 1008$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | ARESFC3 | ARESFC2 | ARESFC1 | ARESFC0 | — | SRESFC4 | SRESFC3 | SRESFC2 | SRESFC1 | SRESFC0 | PRESFC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.7 Reset Factor Clear Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 10 | ARESFC3 | HW BIST Execution Flag Clear<br>0: —<br>1: Clear flag<br>This bit is read as 0 |
| 9 | ARESFC2 | ECM Application Reset Flag Clear<br>0: —<br>1: Clear reset flag<br>This bit is read as 0 |
| 8 | ARESFC1 | Software Application Reset 1 Flag Clear<br>0: —<br>1: Clear reset flag<br>This bit is read as 0 |
| 7 | ARESFC0 | Software Application Reset 0 Flag Clear<br>0: —<br>1: Clear reset flag<br>This bit is read as 0 |
| 6 | Reserved | When read, the value after reset is read. when writing, write the “1”. |
| 5 | SRESFC4 | ECM System Reset Flag Clear<br>0: —<br>1: Clear reset flag<br>This bit is read as 0 |
| 4 | SRESFC3 | Software System Reset 1 Flag Clear<br>0: —<br>1: Clear reset flag<br>This bit is read as 0 |
| 3 | SRESFC2 | Software System Reset 0 Flag Clear<br>0: —<br>1: Clear reset flag<br>This bit is read as 0 |
| 2 | SRESFC1 | CVM Reset Flag Clear<br>0: —<br>1: Clear reset flag<br>This bit is read as 0 |

Table 8.7 Reset Factor Clear Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | SRESFC0 | Terminal Reset Flag Clear<br>0: —<br>1: Clear reset flag<br>This bit is read as 0 |
| 0 | PRESFC0 | Power On Reset Flag Clear<br>0: —<br>1: Clear reset flag<br>This bit is read as 0 |

### 8.3.3 ICUMRESF — ICUMC Reset Factor Register

This register indicates whether a reset occurred since it was cleared last time.

All bits will be cleared by a Power On Reset, CVM Reset or by software.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 6300$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ICUMLRESF1 | ICUMLRESF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.8 ICUMC Reset Factor Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. |
| 1 | ICUMLRESF1 | ICUMC Software Reset Flag<br>0: No reset occurred<br>1: Reset has occurred |
| 0 | ICUMLRESF0 | ICUMC WDTA Reset Flag<br>0: No reset occurred<br>1: Reset has occurred |

### 8.3.4 ICUMRESFC — ICUMC Reset Factor Clear Register

This register clears the reset flags of the ICUMRESF register.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 6308$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ICUMLRESFC1 | ICUMLRESFC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.9 ICUMC Reset Factor Clear Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | ICUMLRESFC1 | ICUMC Software Reset Flag Clear<br>0: —<br>1: Clear reset flag<br>This bit is read as 0. |
| 0 | ICUMLRESFC0 | ICUMC WDTA Reset Flag Clear<br>0: —<br>1: Clear reset flag<br>This bit is read as 0. |

### 8.3.5 SWSRESA0 — Software System Reset Assertion Register 0

This register is used to generate a System Reset 2.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 $1100_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWSRESA0_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.10 Software System Reset Assertion Register 0 Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWSRESA0_0 | Software System Reset Trigger 0<br>This bit is read as 0.<br>0: no function<br>1: generate software system reset 2 |

### 8.3.6 SWSRESA1 — Software System Reset Assertion Register 1

This register is used to generate a System Reset 2 by ICUMC.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 1200$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWSRESA1_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.11 Software System Reset Assertion Register 1 Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWSRESA1_0 | Software System Reset Trigger 1<br>This bit is read as 0.<br>0: no function<br>1: generate software system reset 2 by ICUMC |

### 8.3.7 SWARESA0 — Software Application Reset Assertion Register 0

This register is used to generate an Application Reset 1.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 $1200_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWARESA0_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.12 Software System Reset Assertion Register 0 Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWARESA0_0 | Software Application Reset Trigger 0<br>This bit is read as 0.<br>0: no function<br>1: generate software application reset 1 |

### 8.3.8 SWARESA1 — Software Application Reset Assertion Register 1

This register is used to generate an Application Reset 1 by ICUMC.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 6100$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWARESA1_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.13 Software Application Reset Assertion Register 1 Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWARESA1_0 | Software Application Reset Trigger 1<br>This bit is read as 0.<br>0: no function<br>1: generate software application reset by ICUMC |

### 8.3.9 SWLRESA2 — Software Limited Reset Assertion Register for ICUMC

This register is used to generate a Limited Reset for ICUMC.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 6208$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESA2_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.14 Software Limited Reset Assertion Register for ICUMC Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWLRESA2_0 | Software Limited Reset Trigger for ICUMC<br>This bit is read as 0.<br>0: no function<br>1: generate limited reset |

### 8.3.10 SWLRESA3 — Software Limited Reset Assertion Register for MCAN

This register is used to generate a Limited Reset for MCAN.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 1708$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESA3_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.15 Software Limited Reset Assertion Register for MCAN Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWLRESA3_0 | Software Limited Reset Trigger for MCAN<br>This bit is read as 0.<br>0: no function<br>1: generate limited reset |

### 8.3.11 SWLRESA4 — Software Limited Reset Assertion Register for FlexRay

This register is used to generate a Limited Reset for FlexRay.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 1808$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESA4_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.16 Software Limited Reset Assertion Register for FlexRay Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWLRESA4_0 | Software Limited Reset Trigger for FlexRay<br>This bit is read as 0.<br>0: no function<br>1: generate limited reset |

### 8.3.12 SWLRESA5 — Software Limited Reset Assertion Register for GTM

This register is used to generate a Limited Reset for GTM.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 1908$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESA5_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.17 Software Limited Reset Assertion Register for GTM Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWLRESA5_0 | Software Limited Reset Trigger for GTM<br>This bit is read as 0.<br>0: no function<br>1: generate limited reset |

### 8.3.13 SWLRESA6 — Software Limited Reset Assertion Register for Ethernet

This register is used to generate a Limited Reset for Ethernet.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 1A08$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESA6_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.18 Software Limited Reset Assertion Register for Ethernet Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWLRESA6_0 | Software Limited Reset Trigger for Ethernet<br>This bit is read as 0.<br>0: no function<br>1: generate limited reset |

### 8.3.14 SWLRESA7 — Software Limited Reset Assertion Register for SENT

This register is used to generate a Limited Reset for SENT.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 1B08$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESA7_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.19 Software Limited Reset Assertion Register for SENT Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWLRESA7_0 | Software Limited Reset Trigger for SENT<br>This bit is read as 0.<br>0: no function<br>1: generate limited reset |

### 8.3.15 SWLRESA8 — Software Limited Reset Assertion Register for HS-USRT

This register is used to generate a Limited Reset for HS-USRT.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 1C08$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESA8_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.20 Software Limited Reset Assertion Register for HS-USRT Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWLRESA8_0 | Software Limited Reset Trigger for HS-USRT<br>This bit is read as 0.<br>0: no function<br>1: generate limited reset |

### 8.3.16 SWLRESA10 — Software Limited Reset Assertion Register for CSIH

This register is used to generate a Limited Reset for CSIH.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 1E08$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESA10_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.21 Software Limited Reset Assertion Register for CSIH Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWLRESA10_0 | Software Limited Reset Trigger for CSIH<br>This bit is read as 0.<br>0: no function<br>1: generate limited reset |

### 8.3.17 SWLRESA11 — Software Limited Reset Assertion Register for RLIN3

This register is used to generate a Limited Reset for RLIN3.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 1F08$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESA11_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.22 Software Limited Reset Assertion Register for RLIN3 Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWLRESA11_0 | Software Limited Reset Trigger for RLIN3<br>This bit is read as 0.<br>0: no function<br>1: generate limited reset |

## 8.3.18 SWLRESA12 — Software Limited Reset Assertion Register for AD

This register is used to generate a Limited Reset for AD.

**Access:** This register can be writtern in 32-bit units.

**Address:** FFF8 2008$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESA12_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 8.23 Software Limited Reset Assertion Register for AD Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | SWLRESA12_0 | Software Limited Reset Trigger for AD<br>This bit is read as 0.<br>0: no function<br>1: generate limited reset |

### 8.3.19 SWLRESS2 — Software Limited Reset Status Register for ICUMC

This register is used to show the execution status of a Limited Reset for ICUMC.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 620C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESS2_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.24 Software Limited Reset Status Register for ICUMC Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SWLRESS2_0 | Software Limited Reset Status for ICUMC<br>0: Reset execution is not being processed.<br>1: Reset execution is being processed. |

## 8.3.20 SWLRESS3 — Software Limited Reset Status Register for MCAN

This register is used to show the execution status of a Limited Reset for MCAN.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 170C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESS3_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.25 Software Limited Reset Status Register for MCAN Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SWLRESS3_0 | Software Limited Reset Status for MCAN<br>0: Reset execution is not being processed.<br>1: Reset execution is being processed. |

### 8.3.21 SWLRESS4 — Software Limited Reset Status Register for FlexRay

This register is used to show the execution status of a Limited Reset for FlexRay.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 180C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESS4_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.26 Software Limited Reset Status Register for FlexRay Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SWLRESS4_0 | Software Limited Reset Status for FlexRay<br>0: Reset execution is not being processed.<br>1: Reset execution is being processed. |

## 8.3.22 SWLRESS5 — Software Limited Reset Status Register for GTM

This register is used to show the execution status of a Limited Reset for GTM.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 190C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESS5_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.27 Software Limited Reset Status Register for GTM Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SWLRESS5_0 | Software Limited Reset Status for GTM<br>0: Reset execution is not being processed.<br>1: Reset execution is being processed. |

### 8.3.23 SWLRESS6 — Software Limited Reset Status Register for Ethernet

This register is used to show the execution status of a Limited Reset for Ethernet.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 1A0C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESS6_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.28 Software Limited Reset Status Register for Ethernet Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SWLRESS6_0 | Software Limited Reset Status for Ethernet<br>0: Reset execution is not being processed.<br>1: Reset execution is being processed. |

### 8.3.24 SWLRESS7 — Software Limited Reset Status Register for SENT

This register is used to show the execution status of a Limited Reset for SENT.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 1B0C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESS7_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.29 Software Limited Reset Status Register for SENT Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SWLRESS7_0 | Software Limited Reset Status for SENT<br>0: Reset execution is not being processed.<br>1: Reset execution is being processed. |

## 8.3.25 SWLRESS8 — Software Limited Reset Status Register for HS-USRT

This register is used to show the execution status of a Limited Reset for HS-USRT.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 1C0C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESS8_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.30 Software Limited Reset Status Register for HS-USRT Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SWLRESS8_0 | Software Limited Reset Status for HS-USRT<br>0: Reset execution is not being processed.<br>1: Reset execution is being processed. |

## 8.3.26 SWLRESS10 — Software Limited Reset Status Register for CSIH

This register is used to show the execution status of a Limited Reset for CSIH.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 1E0C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESS10_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.31 Software Limited Reset Status Register for CSIH Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SWLRESS10_0 | Software Limited Reset Status for CSIH<br>0: Reset execution is not being processed.<br>1: Reset execution is being processed. |

## 8.3.27 SWLRESS11 — Software Limited Reset Status Register for RLIN3

This register is used to show the execution status of a Limited Reset for RLIN3.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 1F0C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESS11_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.32 Software Limited Reset Status Register for RLIN3 Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SWLRESS11_0 | Software Limited Reset Status for RLIN3<br>0: Reset execution is not being processed.<br>1: Reset execution is being processed. |

## 8.3.28 SWLRESS12 — Software Limited Reset Status Register for AD

This register is used to show the execution status of a Limited Reset for AD.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 200C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWLRESS12_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 8.33 Software Limited Reset Status Register for AD Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SWLRESS12_0 | Software Limited Reset Status for AD<br>0: Reset execution is not being processed.<br>1: Reset execution is being processed. |

### 8.3.29 RESC — Reset Configuration Register

This register contains configuration settings for the behavior of the device during reset.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 2800$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RESC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 8.34 Reset Configuration Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | RESC0 | ECM Reset Configuration<br>0: ECM Reset Generates a System Reset 2<br>1: ECM Reset Generates an Application Reset 1 |

### 8.3.30 BOOTCTRL — Boot Control Register

This register controls the Start-Up of each PE (including ICUMC). At the time of reset (Power On Reset, System Reset 1/2, Application Reset 1) release, the PEs are started according to the initial setting of this register, which is defined by a FLASH option byte. After reset release, PE can be started selectively by asserting the corresponding bit of this register. Only 1-write is possible for these bits. 0-write is ignored. See **Table 8.45** for details.
In P1M-C, BOOTCTRL.BC2 can be accessed, but the function is disabled. Initial value of BOOTCTRL.BC2 is defined by a FLASH option byte.

**Access:** This register can be read/write in 32-, 16- or 8-bit units.

**Address:** FFC5 $8000_H$

**Value after reset:** 0000 $000X_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | BC2 | BC1 | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R |

**Table 8.35 Boot Control Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4, 3 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 2 | BC2 | Boot Control 2<br>This bit triggers the start-up of the PE2<br>0: Inactive<br>1: Active<br><br>In P1M-C,<br>x: Disable |
| 1 | BC1 | Boot Control 1<br>This bit triggers the start-up of the PE1<br>0: Inactive<br>1: Active |
| 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### 8.3.31 STAC_DTSRAM — RAM Initialization Mode Control Register for DTS_RAM

This register is used to control the RAM initialization execution of the DTSRAM. In Application Reset 1, initialization of DTS_RAM is executed depending on this register.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 $1320_H$

**Value after reset:** 0000 $0003_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RZERO MD1 | RZERO MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 8.36 RAM Initialization Mode Control Register for DTS_RAM Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | RZEROMD[1:0] | RAM Initialization Mode for DTSRAM<br>x0: Disabled<br>01: Prohibited<br>11: Enabled |

### 8.3.32 STAC_GRAM — RAM Initialization Mode Control Register for GRAM

This register is used to control the RAM initialization execution of the GRAM. In Application Reset 1, initialization of GRAM is executed depending on this register.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 $1420_H$

**Value after reset:** 0000 $0003_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RZERO MD1 | RZERO MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 8.37 RAM Initialization Mode Control Register for GRAM Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | RZEROMD[1:0] | RAM Initialization Mode for GRAM<br>x0: Disabled<br>01: Prohibited<br>11: Enabled |

### 8.3.33 STAC_LM0 — RAM Initialization Mode Control Register for PE1

This register is used to control the RAM initialization execution of the PE1 LRAM. In Application Reset 1, initialization of PE1 LRAM is executed depending on this register.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 $1520_H$

**Value after reset:** 0000 $0003_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RZERO MD1 | RZERO MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 8.38 RAM Initialization Mode Control Register for PE1 Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | RZEROMD[1:0] | RAM Initialization Mode for PE1 LRAM<br>x0: Disabled<br>01: Enabled (PE1 Local RAM excluding Backup Section)<br>11: Enabled (PE1 Local RAM including Backup Section) |

### 8.3.34 STAC_LM1 — RAM Initialization Mode Control Register for PE2

This register is used to control the RAM initialization execution of the PE2 LRAM. In Application Reset 1, initialization of PE2 LRAM is executed depending on this register.
In P1M-C, this register can be accessed, but the function is disabled.

**Access:** This register can be read/write in 32-bit units.
**Address:** FFF8 $1620_H$
**Value after reset:** 0000 $0003_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RZEROMD1 | RZEROMD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 8.39 RAM Initialization Mode Control Register for PE2 Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are read as 0. The write value should be 0. |
| 1, 0 | RZEROMD[1:0] | RAM Initialization Mode for PE2 LRAM (In P1H-C and P1H-CE)<br>x0: Disabled<br>01: Prohibited<br>11: Enabled<br><br>In P1M-C,<br>xx: Disabled |

### 8.3.35 STAC_LM2 — RAM Initialization Mode Control Register for ICUMC

This register is used to control the RAM initialization execution of the ICUMC LRAM. In Application Reset 1 and Limited Reset of ICUMC, initialization of ICUMC LRAM is executed depending on this register.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 6220$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RZEROMD1 | RZEROMD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 8.40 RAM Initialization Mode Control Register for ICUMC Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | RZEROMD[1:0] | RAM Initialization Mode for ICUMC LRAM<br>x0: Disabled<br>01: Prohibited<br>11: Enabled |

### 8.3.36 STAC_LM3 — RAM Initialization Mode Control Register for MCAN

This register is used to control the RAM initialization execution of the MCAN. In Application Reset 1 and Limited Reset of MCAN, initialization of RAM for MCAN is executed depending on this register.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 1720$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RZERO MD1 | RZERO MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 8.41 RAM Initialization Mode Control Register for MCAN Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | RZEROMD[1:0] | RAM Initialization Mode for MCAN<br>x0: Disabled<br>01: Prohibited<br>11: Enabled |

If MCAN is disabled by the Flash option, initialization of RAM for MCAN is not executed regardless of this register.

### 8.3.37 STAC_LM4 — RAM Initialization Mode Control Register for FlexRay

This register is used to control the RAM initialization execution of the FlexRay. In Application Reset 1 and Limited Reset of FlexRay, initialization of RAM for FlexRay is executed depending on this register.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 $1820_H$

**Value after reset:** $0000\ 0003_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RZEROMD1 | RZEROMD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 8.42 RAM Initialization Mode Control Register for FlexRay Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | RZEROMD[1:0] | RAM initialization Mode for FlexRay<br>x0: Disabled<br>01: Prohibited<br>11: Enabled |

If FlexRay is disabled by the Flash option, initialization of RAM for FlexRay is not executed regardless of this register.

### 8.3.38 STAC_LM5 — RAM Initialization Mode Control Register for GTM

This register is used to control the RAM initialization execution of the GTM. In Application Reset 1 and Limited Reset of GTM, initialization of RAM for GTM is executed depending on this register.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 1920$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RZERO MD1 | RZERO MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 8.43 RAM Initialization Mode Control Register for GTM Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | RZEROMD[1:0] | RAM initialization Mode for GTM<br>x0: Disabled<br>01: Prohibited<br>11: Enabled |

If GTM is disabled by the Flash option, initialization of RAM for GTM is not executed regardless of this register.

### 8.3.39 STAC_LM10 — RAM Initialization Mode Control Register for CSIH

This register is used to control the RAM Initialization execution of the CSIH. In Application Reset 1 and Limited Reset of CSIH, initialization of RAM for CSIH is executed depending on this register.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 1E20$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RZERO MD1 | RZERO MD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 8.44 RAM Initialization Mode Control Register for CSIH Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | RZEROMD[1:0] | RAM Initialization Mode for CSIH<br>x0: Disabled<br>01: Prohibited<br>11: Enabled |

## 8.4 Operation

### 8.4.1 Reset Categories

#### Power On Reset

The Power On Reset represents a Start-Up of the device during power up. This reset occurs when SYSVCC supply voltage is under a definite level and is released when SYSVCC supply voltage is over the definite level. In this case entire microcontroller is initialized.

#### System Reset 1

System Reset 1 is identical to the Power On Reset with the following exceptions. The Reset Factor Register of the Reset Controller and the Error Source Status Register of the Error Control Module and a part of Core Voltage Monitor registers are excluded from this reset. System Reset 1 sources are as follows.

- Terminal Reset
- Core Voltage Monitor Reset

#### System Reset 2

System Reset 2 is identical to the System Reset 1 with the following exceptions. A part of the Clock Controller registers is preserved (Divider0/1, Selector0). System Reset 2 sources are as follows.

- Software Reset by writing the associated register
- Error Control Module Reset ( The Reset Category is configured by the Reset Configuration Register )

#### Application Reset 1

Application Reset 1 is used for fast re-initialization of the application. This reset is basically identical to the System Reset 2 with the following exceptions. The configuration data stored in the FLASH is not reloaded. HW BIST is not executed too. Therefore, fast re-initialization is realized. In addition, reset to the system timer is masked depending on a register setting. Application Reset 1 sources are as follows.

- Software Reset by writing the associated register
- Error Control Module Reset (The Reset Category is configured by the Reset Configuration Register)

#### Limited Reset

Limited Reset is used for re-initialization of dedicated device function like certain peripherals and the ICUMC. PE1 and PE2 does’t have Limited Reset function.

## 8.4.2 Reset Sources

### Power On Reset

The Power On Reset identifies a Start-Up of the device during power up. The power up condition is detected by a Power-On-Clear circuit (POC). It permanently compares the power supply voltage SYSVCC with an internal reference voltage (Vpoc). If SYSVCC lowers below the internal reference voltate (SYSVCC < Vpoc), Power On Reset is generated.

### Terminal Reset (by RESETZ)

A dedicated Reset Input pin (RESETZ) is available for initialization.

The associated buffer has input hysteresis. Therefore, there is no restriction for the slew rate of the reset signal as defined in the Electrical Characteristics.

The Terminal Reset is active low. Its status will propagate independently from clock activity.

In case of an open reset input, the device will be initialized via the internal pull-down of the Reset Input Buffer. This ensures that the device is always in a safe state.

A Reset glitch filter is provided.

In case Power-up, RESETZ must be asserted until VDD exceeds the threshold and main oscillator is stabilized. Until SYSVCC and VDD exceed the threshold, Internal reset is generated. After Internal reset and RESETZ are released, HW BIST and RAM initialization are executed. For details on the required RESETZ low period (tDVDDPUR, tDVCCPUR) and the threshold of SYSVCC and VDD, see the "**Section 35, Electrical Specifications**".

In case of Terminal Reset excluding power-up/down, the low pulse width of the RESETZ input must be greater than the value of noise-filter characteristic (tWRSL) to activate the reset.

### Core Voltage Monitor Reset

The Core Voltage Monitor reset can be asserted as soon as one of the core supply voltages is outside the operating range. The execution of the CVM reset can be masked as well. For details , see the **Section 10, Core Voltage Monitor (CVM)**.

The CVM operation is independent from the Terminal Reset.

### ECM Reset

No device failure (e.g. illegal access) shall result in a reset. All failures which are detectable by the Error Control Module (ECM) generate dedicated internal reset or interrupts (FENMI or EINT)instead. The failure signals and system behavior can be configured in the ECM (see **Section 29, Error Control Module (ECM)**). Optionally the device failures can be configured to be used as a Reset Source.

The ECM Reset can generate a System Reset 2 or an Application Reset 1. The behavior can be configured by the Reset Configuration Register.

### Software Reset

The device supports multiple software resets that can trigger the reset of dedicated Reset Domains:

- Software System Reset Assertion Register 0 (SWSRESA0) triggers a System Reset 2
- Software Application Reset Assertion Register 0 (SWARESA0) triggers an Application Reset 1
- Software Limited Reset Assertion Register n triggers each of the peripheral Reset Domains (Limited Reset).
- Software System Reset Assertion Register 1 (SWSRESA1) for ICUMC(if present) triggers a System Reset 2.
- Software Application Reset Assertion Register 1 (SWARESA1) for ICUMC(if present) triggers an Application Reset 1.

Each software reset can be triggered when software writes to the associated Reset Assertion Register.

These registers are implemented as TMR to avoid reset reaction on single bit fail.

Each Reset Assertion Register is protected against unintended access by configuration of the P-Bus Guards. For details, see **Section 28, Functional Safety**. In addition, the P-Bus Guard allows restricting the register access to certain CPUs.

For example, this feature can be used to:

- prevent the ICUMC from writing to the SWSRESA0/SWARESA0 assigned to the other PEs.
- prevent any PE except of the ICUMC from writing the SWSRESA1/SWARESA1 assigned to the ICUMC

Before writing to limited reset assertion register, please verify that the corresponding peripheral module is in the idle state and other Limited Reset is not being executed. The procedure of Limited reset execution is shown in **Figure 8.1**. For the procedure of checking idle state, see the section of corresponding peripheral module. After limited reset, the reset module excluding ICUMC moves to the module standby state.

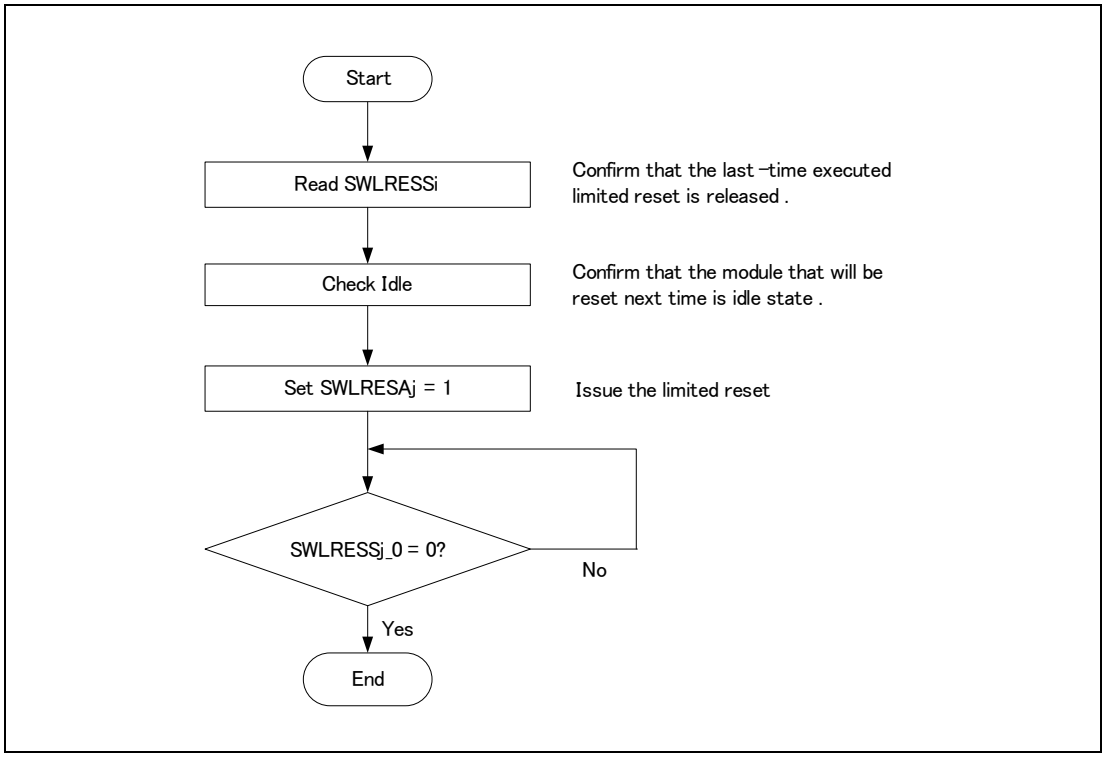



**Figure 8.1 Flowchart of limited reset**

#### ICUMC WDTA Reset

The ICUP Window Watchdog Timer allows interrupting or resetting the ICUMC sub-system in the event of system deadlocks.

### 8.4.3 Reset Flags

Any reset root cause can be identified by SW in the Reset Factor Register(RESF) and the ICUMC Reset Factor Regester (ICUMRESF). Each reset source will be indicated by one bit when it occurs. The status can be read out after the reset has been executed.

### 8.4.4 Read Configuration Data from FLASH

Reading configuration data from FLASH memory is executed by Power On Reset, System Reset 1 and System Reset2. If an ECC Error is detected during reading the configulation data from Flash in Normal Operation Mode, the device will be kept in reset state and will not start up.

### 8.4.5 HW BIST

HW BIST is executed by Power-On-Reset, System Reset 1 and System Reset 2. In System Reset 2, HW BIST execution can be disabled depending on Field BIST control register (BSEQ0CTL). HW BIST isn't executed except in Normal Operation Mode.

### 8.4.6 RAM initialization

RAM initialization to 0 is basically executed by all reset categories. In Application Reset 1 and Limited Reset, RAM initialization can be disabled depending on RAM Initialization Mode Control Register.

### 8.4.7 Start Up of Cores

After reset (Power On Reset , System Reset 1/2,Application Reset 1), start-up cores can be selected by FLASH option byte(STARTUPPE).

The relation of the start-up cores and the FLASH option byte is shown in **Table 8.45**.

In RH850/P1x-C, only PE1 and ICUMC can be selected to start first. ICUMC always must be started first and it is also possible to start PE1 and ICUMC in parallel either .In this case, it is ensured by hardware, that these cores start executing their code simultaneously. About the start-condition of ICUMC, see *the RH850/P1x-C User's Manual: Hardware (Security)* ***Section 3 ICUMC***.

**Table 8.45 Start Up of Cores**

| | The setting of Flash option bytes | | The CPUs to be started first | | |
|---|---|---|---|---|---|
| **Operating Mode** | **STARTUPPE** | **ICUMC Start Enable** | **PE1** | **PE2** | **ICUMC** |
| Normal Operating Mode | 1 | 0 | √ | — | — |
| | 0 | 1 | — | — | √ |
| | 1 | 1 | √ | — | √ |

Another setting except for the above is prohibited.

The CPUs that are started first can trigger the start-up of the others by writing the Boot Control Register (BOOTCTRL). But ICUMC can't be started by SW because ICUMC always starts-up first.

### 8.4.8 Reset Mask function

In debug mode System Reset 1/2 and Application Reset 1 can be masked by debugger setting. For details , see the "Section 31, On-chip Debug Unit (OCD)".

Masked reset sources are shown in **Table 8.46**.

**Table 8.46 Masked reset sources**

| Reset Categories | Reset Sources |
|---|---|
| System Reset 1 | Terminal Reset |
| System Reset 2 | ECM Reset (RESC0 = 0) |
| | Software Reset by SWSRESA0 |
| | Software Reset by SWSRESA1 |
| Application Reset 1 | ECM Reset (RESC0 = 1) |
| | Software Reset by SWARESA0 |
| | Software Reset by SWARESA1 |

### 8.4.9 Reset Output (RESETOUTZ)

A dedicated Reset Output pin RESETOUTZ is available for controlling the reset of external devices. It will be asserted by Power On Reset, System Reset 1, System Reset 2 and Application Reset. After the reset release, its status will be kept and it operates as an Open-Drain (Low) GPIO. It can be released or asserted by writing to the corresponding GPIO control registers.

# Section 9 Power Supply

## 9.1 Features

This section describes the external voltage connection and internal voltage distribution required to operate the microcontroller. The power supply circuit has a POC (Power On Clear) circuit for safe startup.

The power supply of this microcontroller family is dual power supply.

## 9.2 External Pin List

**Table 9.1 The list of external pins** and **Table 9.2 The list of external pins (P1H-CE only)** show the list of external pins.

**Table 9.1 The list of external pins**

| Pin name | I/O | Range | Function |
|---|---|---|---|
| E0VCC | Power | 3.0 to 3.6 V | I/O power supply |
| E1VCC | | | |
| E0VSS | Ground | 0V | |
| E1VSS | | | |
| SYSVCC | Power | 3.0 to 3.6 V | Power supply for IPs (CVM and OSC) |
| VCC | Power | 3.0 to 3.6 V | Internal voltage regulator power |
| VDD | Power | 1.20 V to 1.35 V | Power supply for Internal |
| VSS | Ground | 0 V | |
| OSCVSS | Ground | 0 V | Power supply for Oscillator |
| A0VCC | Power | 3.0 V to 3.6 V | Power supply for ADCF |
| A1VCC | | | |
| A0VSS | Ground | 0 V | |
| A1VSS | | | |
| A0VREFH | Input | 3.0 V to 3.6 V | Reference voltage for ADCF |
| A1VREFH | | | |

**Table 9.2 The list of external pins (P1H-CE only)**

| Pin name | I/O | Range | Function |
|---|---|---|---|
| DVCC | Power | 3.0 V to 3.6 V | Power supply for fast trace I/F |
| DVDD | Power | 1.175 V to 1.325 V | |
| DVSS | Ground | 0 V | |
| EMUVCC | Power | 3.0 V to 3.6 V | Power supply for EMU |
| EMUVDD | Power | 1.175 V to 1.325 V | |
| EMUVSS | Ground | 0 V | |
| ERAM0VCC | Power | 3.0 V to 3.6 V | Power supply for ERAM |
| ERAM1VCC | | | |
| ERAMVDD | Power | 1.20 V to 1.35 V | |
| ERAM0VSS | Ground | 0 V | |
| ERAM1VSS | | | |

## 9.3 Block Diagram

### 9.3.1 P1M-C (Dual power supply)

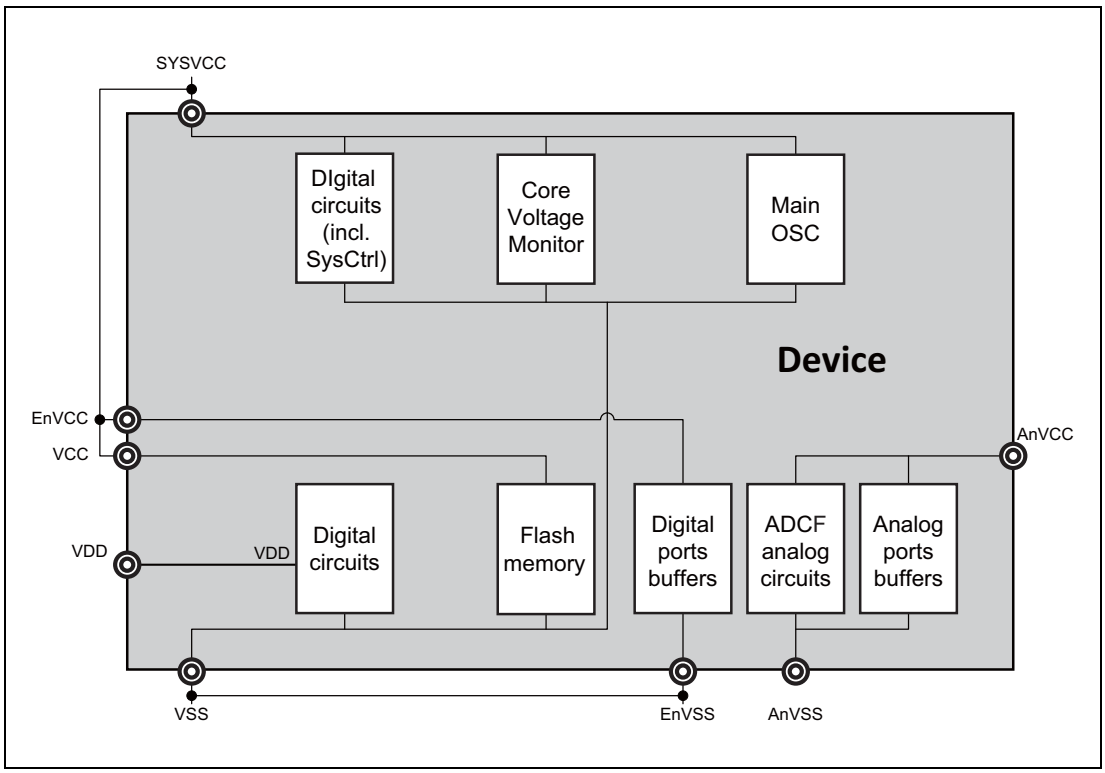



Figure 9.1

### 9.3.2 P1H-C and P1H-CE (Dual power supply)

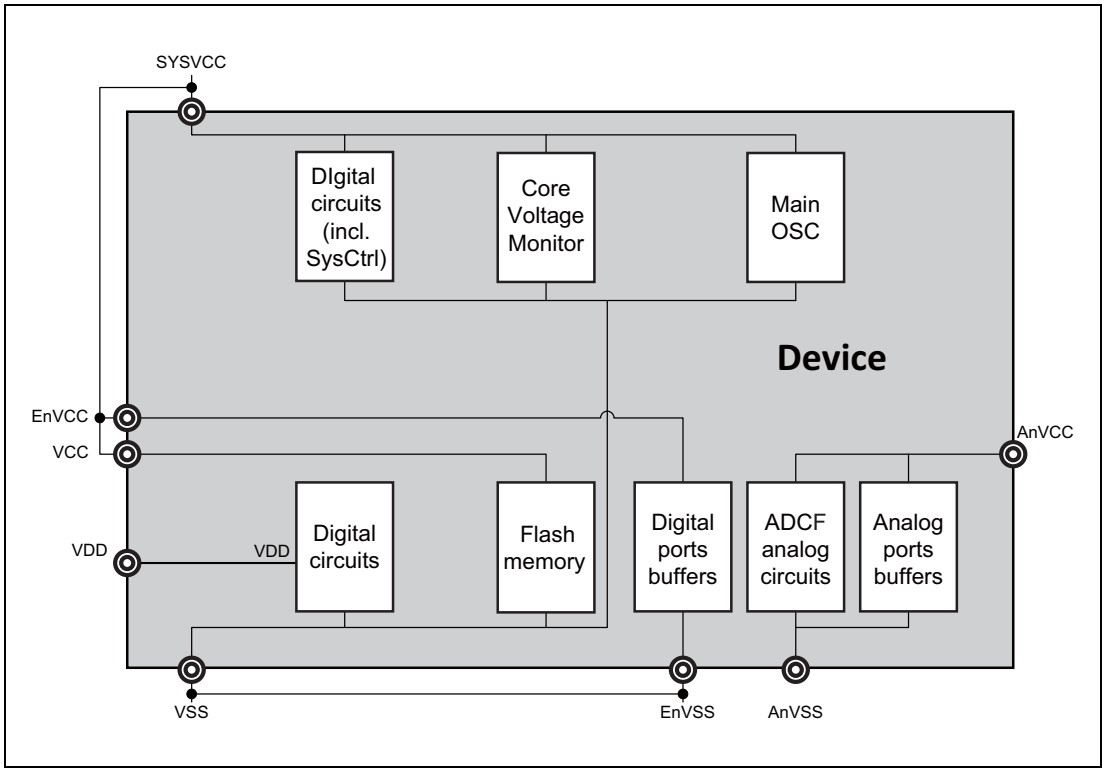



Figure 9.2

## 9.4 Connection Example

### 9.4.1 Dual Power Supply

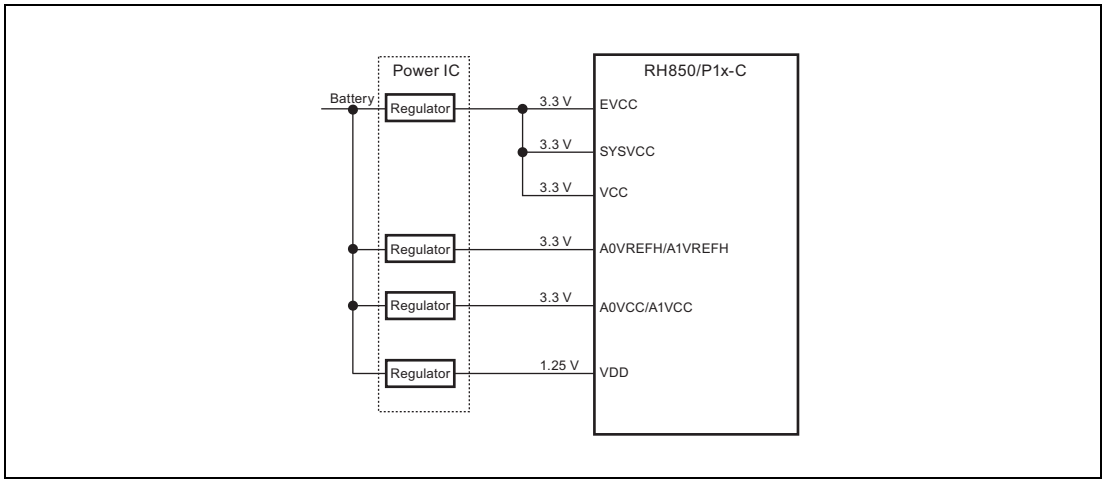



Figure 9.3

## 9.5 Power Up/Down Timing

See **Section 35, Electrical Specifications**.

# Section 10 Core Voltage Monitor (CVM)

## 10.1 Overview of RH850/P1x-C CVM

### 10.1.1 Units

The P1x-C has the following number of units of the CVM.

**Table 10.1 Units**

| Products | P1M-C, P1H-C, P1H-CE |
|---|---|
| Number of units | 1 |
| Name | CVM |

### 10.1.2 Register Addresses

CVM register addresses are represented by an offset from the base address <CVMn_base>.

The following table shows the base address <CVMn_base> of CVM module

**Table 10.2 Register Base Address**

| CVMn | <CVMn_base> Address |
|---|---|
| CVM0 | FFF8 $2000_H$ |

### 10.1.3 Clock Supply

Clock supply by and to a CVM is listed in the following table.

**Table 10.3 Clock Supply**

| Unit Name | Specification | Description |
|---|---|---|
| CVM | Low speed system clock (CLK_LSB) | P-BUS clock for module register |

**Note:** A VDD independent internal OSC is used for the noise filtering.
Refer to **Section 10.4.1**, **(5) Digital noise filter**.

### 10.1.4 Input/Output Pins

I/O pin related to CVM is shown in **Table 10.4**.

**Table 10.4 I/O pins**

| Pin function name | Direction | Description |
|---|---|---|
| CVMOUTZ | Output | CVM detection output |

## 10.2 Overview

The Core voltage monitor (CVM) monitors over and under voltage of the core voltage (VDD).

Violating the operating range of the core voltage is informed by:

- Single output pin (CVMOUTZ).
- Over and under voltage flags
- CVM Reset.

The CVM function is independent of internal or external RESET, except

  – Power On Reset
  – System Reset 1 while diagnostic function.

All writable CVM control registers are protected by a P-Bus guard.

The CVM has a diagnostic function:

- CVM function is testable.
- Over and under-voltage error can be generated without influencing core voltage itself.
- CVM error test is done by changing reference voltage.
- For diagnostic the signal path to the CVMOUTZ pin can be masked.
- CVMOUTZ pin provides read back function to check pin level.

### 10.2.1 Block Diagram

The block diagram of CVM for P1M-C, P1H-C and P1H-CE is shown in **Figure 10.1**.

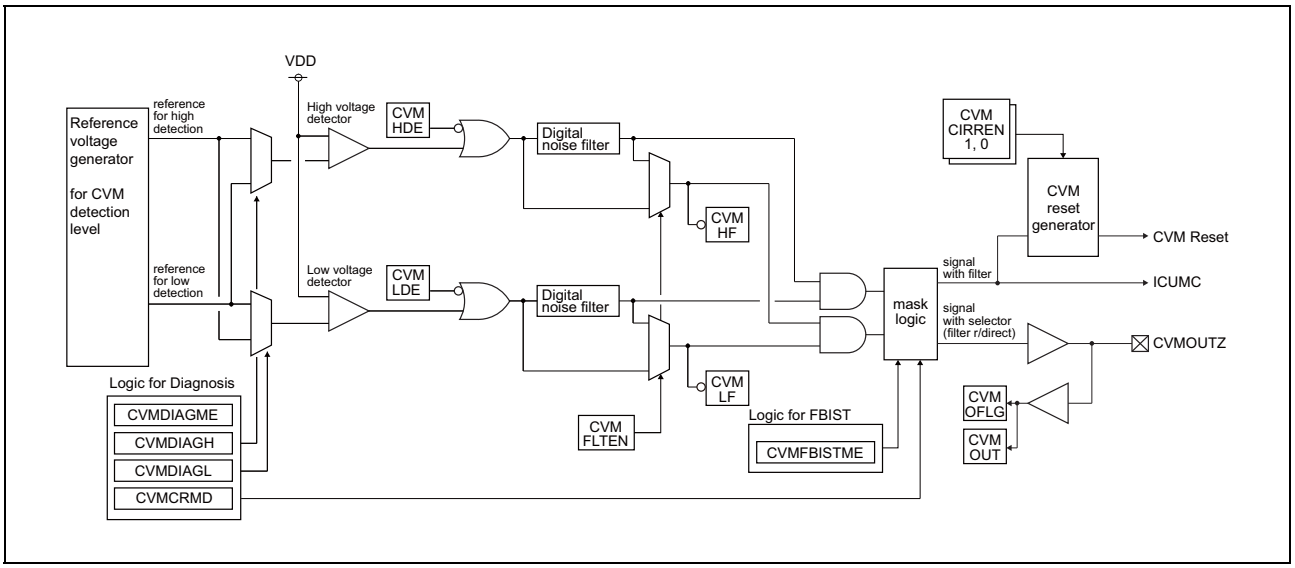



**Figure 10.1 CVM Block diagram for P1M-C, P1H-C and P1H-CE**

# 10.3 Registers

## 10.3.1 List of Registers

A register list related to CVM is shown in **Table 10.5**.

**Table 10.5 List of Registers**

| Address | Register Name | Description | Access Width | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | other |
| <CVM0_base>+C00$_H$ | CVMF | CVM Factor register | 8 | 00$_H$ | PBG4#0. PG4-SC3 | — |
| <CVM0_base>+C04$_H$ | CVMDE | CVM Detection Enable register | 8 | 0x$_H$ | PBG4#0. PG4-SC3 | — |
| <CVM0_base>+C0C$_H$ | CVMDMASK | CVM Detection output diagnosis MASK register | 8 | 00$_H$ | PBG4#0. PG4-SC3 | — |
| <CVM0_base>+C10$_H$ | CVMDIAG | CVM DIAG mode setting register | 8 | 00$_H$ | PBG4#0. PG4-SC3 | — |
| <CVM0_base>+C14$_H$ | CVMMON | CVM Monitor register | 8 | 0x$_H$ *2 | PBG4#0. PG4-SC3 | — |
| <CVM0_base>+C18$_H$ | CVMFC | CVMF clear register | 8 | 00$_H$ | PBG4#0. PG4-SC3 | — |
| <CVM0_base>+C1C$_H$ | CVMDEW | CVMDE write register | 8 | 00$_H$ | PBG4#0. PG4-SC3 | *1 |

Note 1. Writing is permitted only once after power on reset.

Note 2. This value after reset depends on the status of CVMOUTZ pin.

### 10.3.2 Reset of Registers

Register reset conditions are shown in **Table 10.6**.

When VDD error occurs (over or under-voltage detection of CVM), all registers operate normally when the terminal reset is performed after the core voltage has recovered.

In case VDD error occurs and CVM reset is performed, all registers operate normal except for CVMDMASK and CVMDIAG.

A VDD error occurrence is indicated by the CVM factor register (CVMF) and is detectable after VDD has recovered and a system reset 1 has performed. The CVMF keeps the status of a VDD error unaffected of the CVM reset, terminal reset or over or under-voltage of VDD.

**Table 10.6 Register reset conditions**

| Register Name | Reset condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 (*1) | System Reset 2 | Application Reset 1 | Limited Reset |
| CVMF | √ | | | | |
| CVMDE | √ | (*2) | | | |
| CVMDMASK | √ | √ | | | |
| CVMDIAG | √ | √ | | | |
| CVMMON | | | | | |
| CVMFC | | | | | |
| CVMDEW | | | | | |

Note 1. CVM Reset is excluded.

Note 2. CVMDE[1:0] (CVMHDE and CVMLDE) are set by Flash option when Power On Reset or Terminal Reset (RESETZ) is released.

### 10.3.3 CVMF — CVM factor register

CVMF register indicates that an error is detected by CVM. Each flag of CVMF can be cleared individually by writing '1' to the corresponding bit in CVMFC.

This register is initialized only by Power On Reset.

**Access:** This register can be read in 8-bit units.

**Address:** <CVM0_base> +$C00_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CVMOFLG | — | — | — | — | — | CVMHVF | CVMLVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 10.7 CVM Factor register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | CVMOFLG | CVMOUTZ flag<br>When CVMOUTZ pin has changed to low level, this flag is set to 1.<br>0: CVMOUTZ doesn't have changed to low level.<br>1: CVMOUTZ has changed to low level. |
| 6 to 2 | Reserved | These bits are read as 0. |
| 1 | CVMHVF | High voltage detection flag<br>0: No high core voltage violation detected<br>1: High core voltage violation occurred |
| 0 | CVMLVF | Low voltage detection flag<br>0: No low core voltage violation detected<br>1: low core voltage violation occurred |

### 10.3.4 CVMFC — CVMF clear register

CVMFC is a register to clear the CVMF register. The read value of this register is always $00_H$.

**Access:** This register can be written in 8-bit units.

**Address:** <CVM0_base> +$C18_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CVMOFLGC | — | — | — | — | — | CVMHVFC | CVMLVFC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | R | R | R | R | R | W | W |

**Table 10.8 CVMF clear register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | CVMOFLGC | Clear CVMOUTZ flag<br>0: writing 0 has no effect<br>1: writing 1 will clear CVMOFLG |
| 6 to 2 | Reserved | These bits are read as 0. The write value should be 0. |
| 1 | CVMHVFC | Clear high voltage detection flag<br>0: writing 0 has no effect<br>1: writing 1 will clear CVMHVF |
| 0 | CVMLVFC | Clear low voltage detection flag<br>0: writing 0 has no effect<br>1: writing 1 will clear CVMLVF |

### 10.3.5 CVMDE — CVM detection enable register

CVMDE is a read only register to inform about status of CVM control. It is a read back path from the CVM configuration about CVM reset, mask of CVM signal and filter of CVM signal.

Bit1,0 is set by flash option byte when Power On Reset or Terminal Reset (RESETZ) is released.

Bit7, 5, 4 to 2 are initialized by Power On Reset and reflect the settings written to the corresponding bits of CVMDEW.

**Access:** This register can be read in 8-bit units.

**Address:** <CVM0_base> +$C04_H$

**Value after reset:** $0X_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CVMCIRREN1 | — | CVMCIRREN0 | CVMFBISTME | CVMFLTEN | CVMDIAGME | CVMHDE | CVMLDE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | X (Flash-opt) | X (Flash-opt) |
| R/W | R | R | R | R | R | R | R | R |

**Table 10.9 CVM detection enable register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | CVMCIRREN1 | Permit CVM reset.<br>CVMCIRREN1, 0<br>$00_B$: CVM internal reset is not permitted<br>$11_B$: CVM internal reset is permitted |
| 6 | Reserved | This bit is read as 0. |
| 5 | CVMCIRREN0 | refer to bit7 (CVMCIRREN1) |
| 4 | CVMFBISTME | Permit CVMOUTZ output mask for FBIST operation<br>0: During FBIST the CVMOUTZ and CVM reset can be masked<br>1: During FBIST the CVMOUTZ and CVM reset cannot be masked |
| 3 | CVMFLTEN | Enable output filter for CVMOUTZ<br>0: Enable output filter for CVMOUTZ<br>1: Disable output filter for CVMOUTZ |
| 2 | CVMDIAGME | Permit DIAG function<br>0: DIAG function of the CVM can be enabled<br>1: DIAG function of the CVM cannot be enabled |
| 1 | CVMHDE | High voltage detection enable<br>0: Disable high voltage detection<br>1: Enable high voltage detection<br>The value after reset of this register can be set by flash option byte (CVMHDETEN) |
| 0 | CVMLDE | Low voltage detection enable<br>0: Disable low voltage detection<br>1: Enable low voltage detection<br>The value after reset of this register can be set by flash option byte (CVMLDETEN) |

**NOTE**

CVM internal reset function and DIAG function should not be enabled simultaneously.

### 10.3.6 CVMDEW — CVMDE write register

CVMDEW is a register to set values of CVMDE. Writing is permitted only once after Power On Reset was released. Subsequent write operation is ignored. The read value of this register is always $00_H$.

**Access:** This register can be write in 8-bit units.

**Address:** <CVM0_base> +$C1C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CVM CIRREN1W | — | CVM CIRREN0W | CVM FBISTMEW | CVMFLTENW | CVMDIAGMEW | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | R | W | W | W | W | R | R |

**Table 10.10 CVMDE write register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | CVMCIRREN1W | The data written in this bit is set to CVMDE.CVMCIRREN1. |
| 6 | Reserved | This bit is read as 0. The write value should be 0. |
| 5 | CVMCIRREN0W | The data written in this bit is set to CVMDE.CVMCIRREN0. |
| 4 | CVMFBISTMEW | The data written in this bit is set to CVMDE.CVMFBISTME. |
| 3 | CVMFLTENW | The data written in this bit is set to CVMDE.CVMFLTEN. |
| 2 | CVMDIAGMEW | The data written in this bit is set to CVMDE.CVMDIAGME. |
| 1, 0 | Reserved | This bit is read as 0. The write value should be 0. |

**NOTE**

CVMDIAGMEW is needed to be set “1” whenever both CVMCIRREN1W and CVMCIRREN0W are set “1”.

### 10.3.7 CVMDMASK — CVM detection output diagnosis mask register

CVMDMASK is a register to mask CVMOUTZ and the interrupt to ICUMC when CVMDE.CVMDIAGME = 0. CVMDMASK's setting is ignored if CVMDE.CVMDIAGME = 1.

Writing is possible without depending on CVMDIAGME's setting.

This register is initialized by Power On Reset or Terminal Reset (RESETZ) .

**Access:** This register can be read/write in 8-bit units.

**Address:** <CVM0_base> +$C0C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CVMCRMD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 10.11 CVM detection output diagnosis mask register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | These bits are read as 0. The write value should be 0. |
| 0 | CVMCRMD | This bit masks CVMOUTZ and the interrupt to ICUMC when CVMDIAGME = 0.<br>0: CVM control signals are not masked.<br>1: CVM control signals are masked.<br>As a result CVMOUTZ is high. And the interrupt and the reset don't occur.<br>Note: The value of CVMDE.CVMCIRREN1,0 is '$00_B$' while the CVM diagnosis application is carried out after Power On Reset. Therefore CVM reset has been already masked by CVMCIRREN1,0 even if CVMCRMD isn't set to '1'. |

### 10.3.8 CVMDIAG — CVM DIAG mode setting register

CVMDIAG forces CVM comparators to output error. This register is valid only in DIAG mode when CVMDE.CVMDIAGME = 0. CVMDIAG's setting is ignored if CVMDE.CVMDIAGME = 1.

Writing is possible without depending on CVMDIAGME's setting.

This register is initialized by Power On Reset or Terminal Reset (RESETZ) .

**Access:** This register can be read/write in 8-bit units.

**Address:** <CVM0_base> +$C10_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | CVMDIAGH | CVMDIAGL | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R | R |

**Table 10.12 CVM diag mode setting register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | These bits are read as 0. The write value should be 0. |
| 3 | CVMDIAGH | This bit can change the detection level of high voltage detector to low level intentionally when CVMDIAGME = 0.<br>0: CVM for high voltage monitors normally.<br>1: CVM for high voltage will detect violation condition because detection level changes. |
| 2 | CVMDIAGL | This bit can change the detection level of low voltage detector to high level intentionally when CVMDIAGME = 0.<br>0: CVM for low voltage monitors normally.<br>1: CVM for low voltage will detect violation condition because detection level changes. |
| 1, 0 | Reserved | These bits are read as 0. The write value should be 0. |

### 10.3.9 CVMMON — CVM Monitor register

CVMMON register reflects the status of CVMOUTZ pin.

**Access:** This register can be read in 8-bit units.

**Address:** <CVM0_base> +$C14_H$

**Value after reset:** $0X_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CVMOUT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X |
| R/W | R | R | R | R | R | R | R | R |

**Table 10.13 CVM monitor register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | These bits are read as 0. |
| 0 | CVMOUT | CVMOUTZ pin level<br>0: CVMOUTZ Output level is at low level.<br>1: CVMOUTZ Output level is at high level. |

## 10.4 Operation

### 10.4.1 CVM basic function

CVM watches VDD voltage all the time except when VCC or EVCC is not stable and Power On Reset occurs. CVM has high and low voltage detection circuit. If it detects a voltage error, it records the error to CVMF register, report the error internally to the ICUMC and notifies the error to the outside via CVMOUTZ pin. Configurable, the CVM can also generate an internal reset (CVM reset) in case of a voltage error.

(1) CVM detection flag

CVM sets '1' to CVMF.CVMHF when VDD becomes higher than high voltage detection level of CVM. Similarly, CVM sets '1' to CVMF.CVMLF when VDD becomes lower than low voltage detection level of CVM. These flags can be cleared to '0' by writing in the corresponding bit of CVMFC (CVMHFC for CVMHF, CVMLFC for CVMLF). These flags are also cleared by power on reset (not cleared by other resets). **Figure 10.2** shows the operation example of CVMHF and CVMLF.

**NOTE**

For voltage detection level, see **Section 35.7.2, Core Voltage Monitor (CVM) Characteristics**.

(2) CVMOUTZ pin

While VDD voltage is higher than high voltage detection level or lower than low voltage detection level of CVM, CVMOUTZ pin outputs low. If CVM doesn't detect voltage error, CVMOUTZ pin outputs high.

To avoid unwanted operation of CVM circuit caused by instable power supply at power up, CVMOUTZ is fixed to low during power on reset period. CVMOUTZ outputs error detection result during other reset period.

The status of CVMOUTZ is readable from CVMF.CVMOFLG or CVMMON.CVMOUT. CVMOFLG is set to '1' when CVMOUTZ pin becomes low. It is cleared to '0' when '1' is written to CVMFC.CVMOFLGC. It is also cleared by Power On Reset (not cleared by other resets). **Figure 10.2** shows the operation example of CVMOUTZ and CVMOFLG.

CVMMON.CVMOUT is used for self-diagnosis function. See **Section 10.4.3, CVM diagnosis function** for further information.

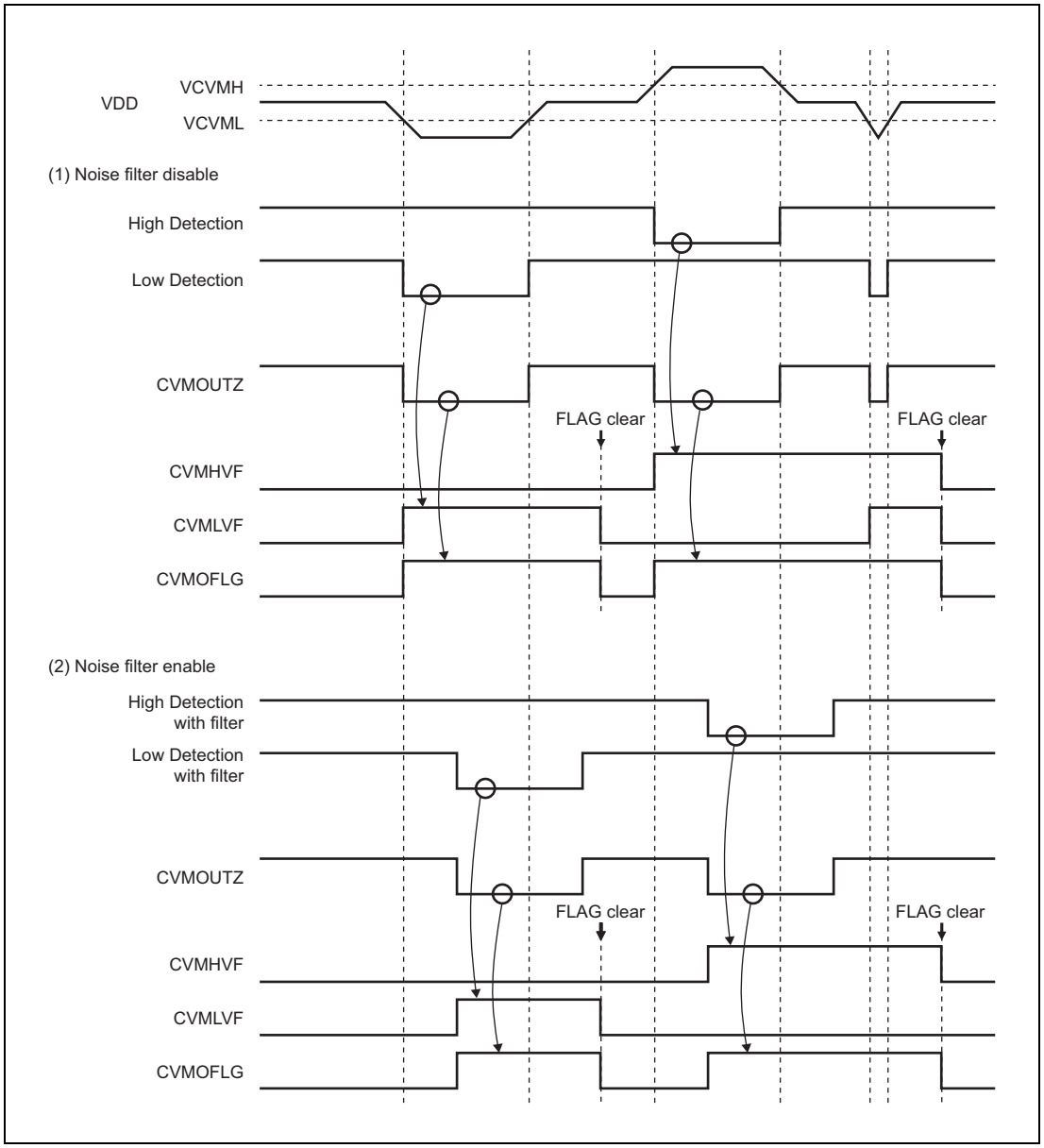



**Figure 10.2 CVM basic operation**

(3) CVM reset

If VDD voltage becomes higher than high voltage detection level or lower than low voltage detection level of CVM when "$11_H$" was set to CVMMDE.CVMCIRREN1, 0 in advance, CVM reset occurs and the MCU is initialized. After CVM detects the normal voltage, CVM reset is released when the time is needed to initialize internal circuits has elapsed. **Figure 10.3** shows the operation example of CVM reset.

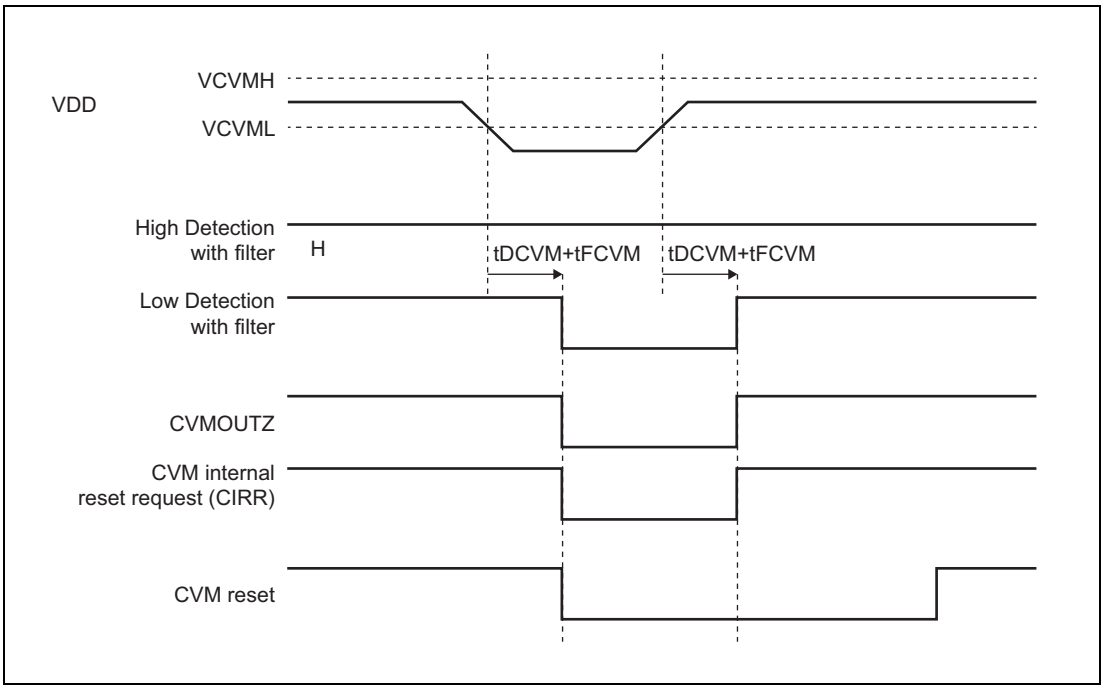



**Figure 10.3 CVM reset operation**

(4) Relation with ICUMC

VDD voltage error detected by CVM is notified to ICUMC. ICUMC can treat this notification as the interrupt if CVM reset is inactivated and VDD is in the operating range

(5) Digital noise filter

The output signal of the CVM comparators passes through a digital noise filter to remove an unintended glitch. Whether to use filtered output or to bypass the filter is selectable by CVMDE.CVMFLTEN. This setting is applied to CVMOUTZ, CVMHF and CVMLF. On the other hand, the filtered signal is always used for CVM reset or the interrupt to ICUMC.

A VDD independent internal OSC is used for the noise filtering.

**NOTE**

For filter timing, see **Section 35.7.2, Core Voltage Monitor (CVM) Characteristics**.

### 10.4.2 CVM function in FBIST and Serial programing mode (Mask of CVMOUTZ pin and CVM reset)

It is possible to mask CVMOUTZ output and the interrupt to ICUMC and CVM reset by CVMDE.CVMFBISTME when field-BIST is carried out after Power On Reset and System Reset1. In this case, CVMOUTZ is fixed to high level. And the request of the interrupt to ICUMC and CVM reset don't occur.

Field-BIST can be executed after system reset2 by setting of BSEQ0CTL.HWBISTEXE bit. However, CVMOUTZ output and the interrupt to ICU-M and CVM reset are not masked when field-BIST is carried out after System Reset2. Therefore, as long as it is known that VDD is stabled within the range when the field-BIST is being executed, field-BIST can be executed after System reset2.

CVMOUTZ output is fixed to high level in Serial programing mode.

### 10.4.3 CVM diagnosis function

(1) Change of CVM detection level for the error injection

When CVMDE.CVMDIAGME is set to '0', CVM detection level can be changed by CVMDIAG.CVMDIAGH or CVMDIAG.CVMDIAGL. CVM error detection signal can be intentionally generated by setting these registers even when VDD voltage is in the operating range.

(2) Mask of CVMOUTZ pin, the interrupt to ICUMC and CVM reset

When CVMDE.CVMDIAGME is set to '0' and CVMDMASK.CVMCRMD is set to '1', CVMOUTZ output and the interrupt to ICUMC can be masked. This setting fixes CVMOUTZ to high level and can avoid the occurrence of the interrupt to ICUMC even if CVM error is occurred by CVMDIAGH or CVMDIAGL.

The value of CVMDE.CVMCIRREN1,0 is '$00_B$' while the CVM diagnosis application is carried out after Power On Reset. Therefore CVM reset has been already masked by CVMCIRREN1,0 even if CVMCRMD isn't set to '1'.

(3) CVMMON register

The status of CVMOUTZ can be monitored by CVMMON.CVMOUT. By reading this register, it can be checked that CVMOUTZ is at low level when CVM error is occurred by CVMDIAGH or CVMDIAGL.

(4) The flow of CVM diagnosis application

CVMDE.CVMDIAGME is initialized to '0' by Power On Reset. So CVM diagnosis function is enabled at start-up after these resets. The user's application controls the diagnostic function by CVMDMASK and CVMDIAG. **Figure 10.4** shows the example of the diagnosis application.

After diagnosis application was over, write '1' to CVMDEW.CVMDIAGMEW*1 and disable CVM diagnosis function immediately. This function protects each flag against erroneous operation when VDD is out of the operating range.

**Note 1.** Set the other bits of CVMDEW too at the same time, because writing in CVMDEW is permitted only once.

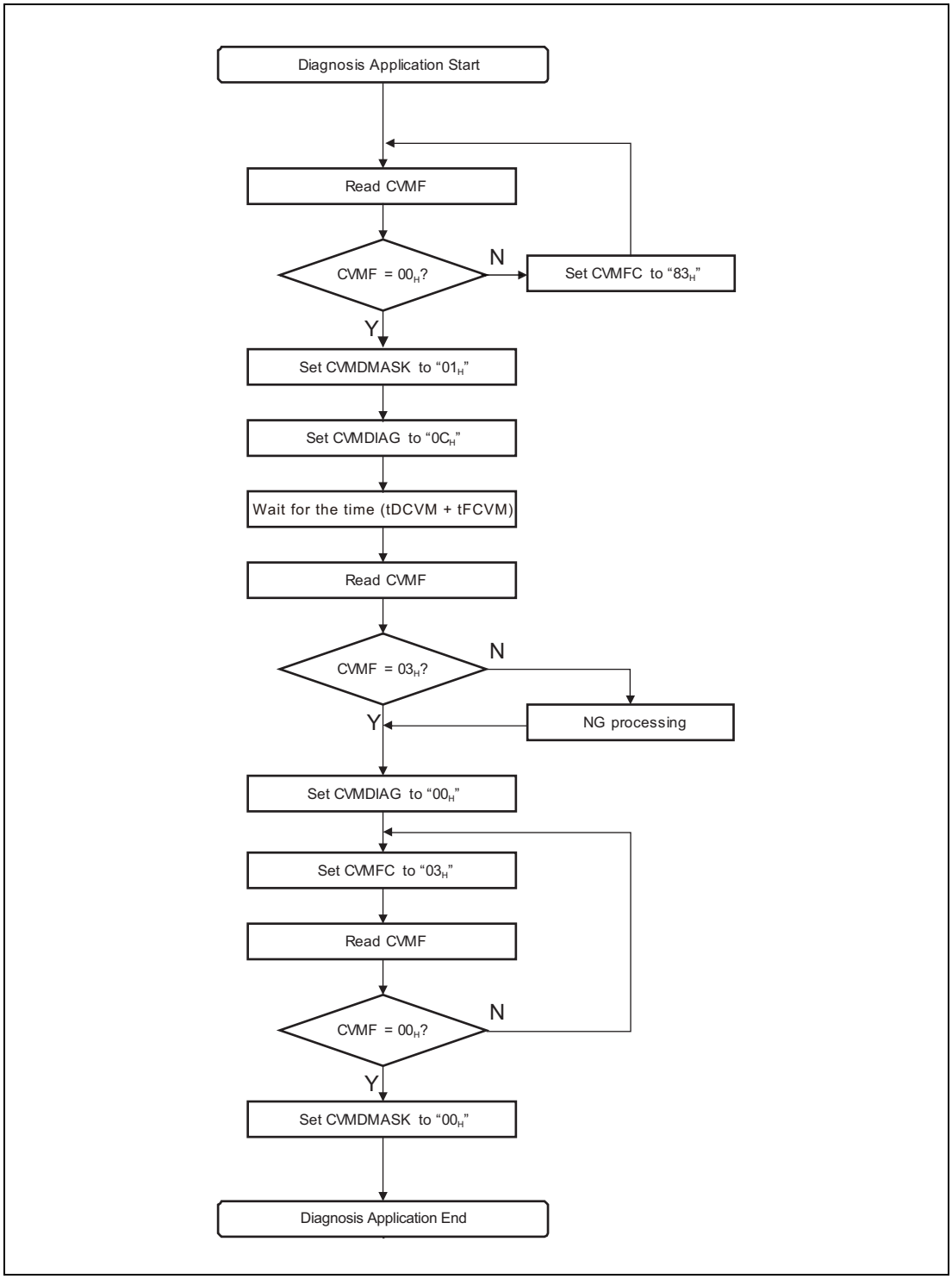



**Figure 10.4 The example of the flowchart of CVM diagnosis application**

## 10.5 Usage Notes

(1) Flag bit (CVMHF, CVMLF, CVMOFLG)

Setting 1 to CVMHF by error detection is prior to clearing CVMHF by CVMHFC. Similarly, setting 1 to CVMLF by error detection is prior to clearing CVMLF by CVMLFC. Setting 1 to CVMOFLG by pin status is prior to clearing CVMOFLG by CVMOFLGC. These functions protect each flag against erroneous operation when VDD is out of the operating range. When VDD is recovered to the operating range, the MCU need to be initialized by RESETZ pin or CVM reset, or the MCU may still continue erroneous operation and then unintended clear of flag may occur.

(2) ERROROUTZ pin

Enable CVM reset if you want to notify CVM error detection signal to outside of the MCU via ERROROUTZ. When CVM reset is occurred, ERROROUTZ is set to Hi-Z (high impedance) regardless of whether VDD is in the operating range or not.

# Section 11 Temperature Sensor

RH850/P1x-C provided with a temperature sensor that outputs digital internal temperature data consists of a sensor section and an separate Delta-Sigma-A/D converter.

## 11.1 Overview of RH850/P1x-C Temperature Sensor

### 11.1.1 Channels

RH850/P1x-C has the following number of channels of temperature sensor.

**Table 11.1 Channels**

| | Temperature Sensor |
|---|---|
| Number of channels | 1 |
| Name | OTS0 |

### 11.1.2 Register Address

The temperature sensor addresses are given as offsets from individual base addresses <OTSn_base>.

OTSn register addresses are listed in the following table.

**Table 11.2 Register Base Address**

| OTSn | <OTSn_base> Address |
|---|---|
| OTS0 | FFF9 $3000_H$ |

### 11.1.3 Clock Supply

Clock supply by and to a temperature sensor is listed in the following table.

**Table 11.3 Clock Supply**

| Unit Name | Clock for the Unit | Supply Clock Name |
|---|---|---|
| OTS0 | P-Bus interface clock (PCLK) | CLK_LSB |
| | Operation clock (OTSCLK) | CLKP_L |

**Note:** The following condition must be fulfilled: 0.6 × CLKP_L ≤ CLK_LSB ≤ CLKP_L
Only when CLKP_L = 36 MHz – 40 MHz, the electrical characteristics can be guaranteed.

### 11.1.4 Interrupt Request

The temperature sensor interrupt requests are listed in the following table.

**Table 11.4 Interrupt Requests**

| Unit Interrupt Name | Outline | Interrupt Number |
|---|---|---|
| OTE | Temperature sensor error interrupt | 37 |
| OTI | Temperature measurement end interrupt | 38 |
| OTULI | Triggered if state machine change the stage by the temperature rising or falling in the guaranteed temperature range | 39 |

**Table 11.5 Alarm Notification**

| Unit Interrupt Name | Outline | Connected to |
|---|---|---|
| OTABE | Abnormal temperature error signal | ECM |

### 11.1.5 Hardware Reset

The registers that constitute the temperature sensor are initialized by the reset sources listed below.

**Table 11.6 Register Reset Condition**

| | | Reset Condition | | | | |
|---|---|---|---|---|---|---|
| Unit Name | Register Name | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| OTS0 | All registers | √ | √ | √ | √ | — |

### 11.1.6 External Input/Output Signals

No external Input and Output pins.

## 11.2 Overview

The following describes features of the temperature sensor.

- Temperature measurement range
  For this manipulation detection the operating temperature borders (TJ, MIN and TJ, MAX) must be precise reachable. Due to delay timing and the limited reachable accuracy the operation of Temperature Sensor itself and ICUMC reaction for thermal attack must exceed the specified range of TJ, MAX. This extended temperature range above TJ, MAX is agreed to be guaranteed by design only and will not be tested.
- Temperature measurement time
  Internal OTS clock (approx. 51.28 kHz) × 512 states
- Temperature data register
  A temperature data register stores a temperature measurement value.
- Temperature measurement mode
  Single measurement mode: Used to measure temperature only once.
  Continuous measurement mode: Used to measure temperature continuously.
- Supporting temperature measurement end interrupt
  Each time temperature measurement ends, the temperature sensor can generate an interrupt request (OTI) to be sent to the INTC.
- Supporting temperature alarm error and temperature rise/drop interrupt
  The temperature sensor sets six temperature threshold values (high-temperature border AH > high-temperature border AL > high-temperature border BH > high-temperature border BL > low-temperature border AH > low-temperature border AL) in advance. Furthermore, the temperature sensor has four temperature states (high temperature A, high temperature B, ordinary temperature, and low temperature A). These temperature states are updated at each temperature measurement according to six temperature threshold values. The following describes conditions for generating a temperature alarm error and a temperature rise/drop interrupt.
  - A temperature alarm error (OTABE) is output at a transition from a state other than high temperature A to high temperature A or at a transition from a state other than low temperature A to low temperature A.
  - A temperature rise/drop interrupt (OTULI) is output at a transition from high temperature A to high temperature B or ordinary temperature or at a transition from high temperature B to ordinary temperature, that is, when temperature drops.
  - A temperature rise/drop interrupt (OTULI) is output at a transition from low temperature A to high temperature B or ordinary temperature or at a transition from ordinary temperature to high temperature B, that is, when temperature rises.

Temperature status can be monitored by reading the temperature status register.

- Diagnosis function
  When temperature is measured in continuous measurement mode, if the difference between the measured value and the immediately previous measured value is larger than the prespecified difference (temperature difference limit value), a temperature sensor error (OTE) is output.
- Reducing errors by temperature correction
  Three temperature correction coefficients are stored beforehand in coefficient registers A to C.

## 11.3 Configuration

**Figure 11.1** shows a block diagram of the temperature sensor.

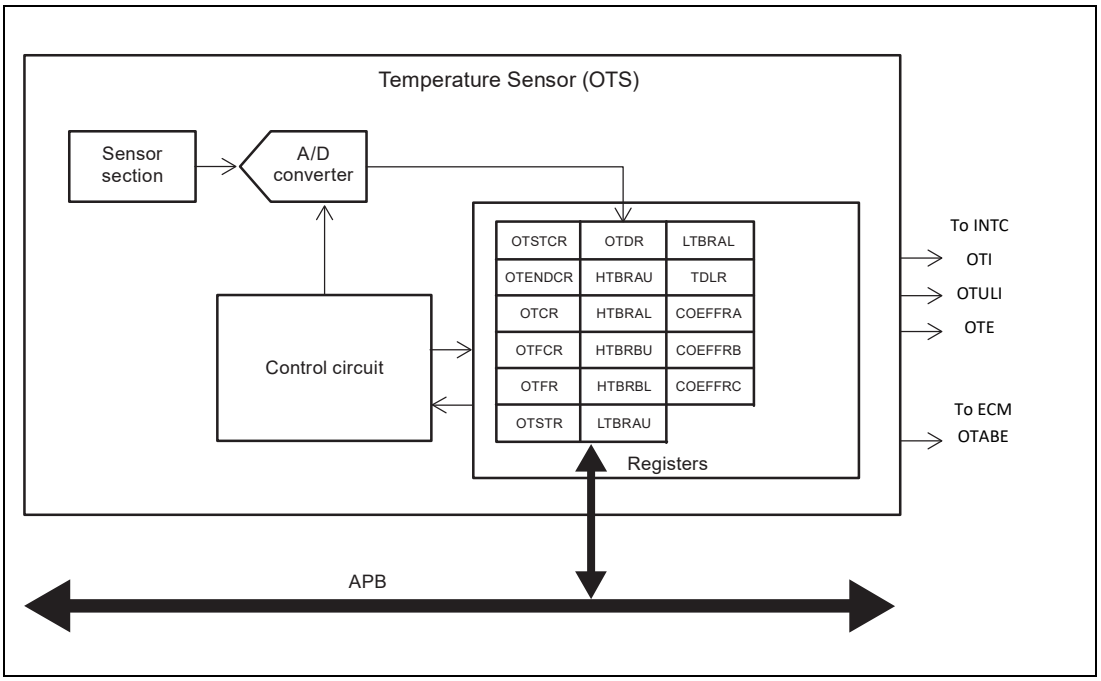



**Figure 11.1 Temperature Sensor Block Diagram**

### 11.3.1 List of Registers

The following table lists registers of the temperature sensor.

| Address Offset | Register Name | Description | Access Width (bits) | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | Other |
| $000_H$ | OTS0OTSTCR | Temperature measurement start control register | 8 | $00_H$ | PBG4#1. PG4-OTS | — |
| $004_H$ | OTS0OTENDCR | Temperature measurement end control register | 8 | $00_H$ | PBG4#1. PG4-OTS | — |
| $008_H$ | OTS0OTCR | Temperature sensor control register | 8 | $00_H$ | PBG4#1. PG4-OTS | — |
| $00C_H$ | OTS0OTFCR | Temperature sensor flag clear register | 8 | $00_H$ | PBG4#1. PG4-OTS | — |
| $010_H$ | OTS0OTFR | Temperature sensor flag register | 8 | $00_H$ | PBG4#1. PG4-OTS | — |
| $014_H$ | OTS0OTSTR | Temperature status register | 8 | $00_H$ | PBG4#1. PG4-OTS | — |
| $018_H$ | OTS0OTDR | Temperature data register | 16 | $0000_H$ | PBG4#1. PG4-OTS | — |
| $01C_H$ | OTS0HTBRAU | High-temperature border AU register | 16 | $0000_H$ | PBG4#1. PG4-OTS | — |
| $020_H$ | OTS0HTBRAL | High-temperature border AL register | 16 | $0000_H$ | PBG4#1. PG4-OTS | — |
| $024_H$ | OTS0HTBRBU | High-temperature border BU register | 16 | $0000_H$ | PBG4#1. PG4-OTS | — |
| $028_H$ | OTS0HTBRBL | High-temperature border BL register | 16 | $0000_H$ | PBG4#1. PG4-OTS | — |
| $02C_H$ | OTS0LTBRAU | Low-temperature border AU register | 16 | $0000_H$ | PBG4#1. PG4-OTS | — |
| $030_H$ | OTS0LTBRAL | Low-temperature border AL register | 16 | $0000_H$ | PBG4#1. PG4-OTS | — |
| $034_H$ | OTS0TDLR | Temperature difference limiting register | 16 | $7FFF_H$ | PBG4#1. PG4-OTS | — |
| $038_H$ | OTS0COEFFRA | Coefficient A register | 16 | N/A | PBG4#1. PG4-OTS | — |
| $03C_H$ | OTS0COEFFRB | Coefficient B register | 16 | N/A | PBG4#1. PG4-OTS | — |
| $040_H$ | OTS0COEFFRC | Coefficient C register | 16 | N/A | PBG4#1. PG4-OTS | — |

## 11.4 Details of Registers

This section describes registers of the temperature sensor.

### 11.4.1 OTS0OTSTCR — Temperature Measurement Start Control Register

OTS0OTSTCR is an 8-bit write-only register to control starting the temperature sensor. This register is always read as 0.

**Access:** This register can be written in 8-bit units.
**Address:** <OTSn_base> + $000_H$
**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | OTST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 11.7 Contents of OTS0OTSTCR Register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | OTST | Temperature Measurement Start<br>Condition for starting temperature measurement by OTST<br>Write 1 to OTST when OTACT = 0 |

### 11.4.2 OTS0OTENDCR — Temperature Measurement End Control Register

OTS0OTENDCR is an 8-bit write-only register to control terminating the temperature sensor. This register is always read as 0.

**Access:** This register can be written in 8-bit units.
**Address:** <OTSn_base> + $004_H$
**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | OTEND |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 11.8 Contents of OTS0OTENDCR Register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to1 | Reserved | When writing, write the value after reset. |
| 0 | OTEND*1 | Temperature Measurement End<br>Condition for terminating temperature measurement by OTEND<br>Write 1 to OTEND and then wait for 2-cycle OTS clock. |

Note 1. When doing stop processing by OTEND bit, the measuring data is broken. When OTEND bit is set to 1 before once of temperature measurement was completed in a once measurement mode, do not output a measurement result, measurement operating is ended.

### 11.4.3 OTS0OTCR — Temperature Sensor Control Register

OTS0OTCR is an 8-bit readable/writable register to control the temperature sensor. OTS0OTCR is initialized to $00_H$ by any reset.

Access: This register can be read/written in 8-bit units.

Address: <OTSn_base> + $008_H$

**Value after reset**: $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | OTEE | OTULIE | OTABEE | SDE | OTMD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 11.9 Contents of OTS0OTCR Register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | OTEE | Temperature Sensor Error Enable<br>0: Disabled<br>1: Enabled<br>When OTMD = 0 (single measurement mode) or SDE = 0 (diagnosis disabled), no temperature sensor error is generated regardless of the OTEE setting. |
| 3 | OTULIE | Temperature Rise /Drop Interrupt Enable<br>0: Disabled<br>1: Enabled |
| 2 | OTABEE | Temperature Alarm Error Enable<br>0: Disabled<br>1: Enabled |
| 1 | SDE | Diagnosis Enable<br>0: Disabled<br>1: Enabled<br>When OTMD = 0 (single measurement mode), diagnosis is not performed regardless of the SDE setting. |
| 0 | OTMD | Temperature Measurement Mode<br>0: Single temperature measurement mode<br>1: Continuous temperature measurement mode |

**CAUTION**

To prevent malfunction, set OTS0OTCR while the OTACT bit in OTS0OTFR is 0.

### 11.4.4 OTS0OTFCR — Temperature Sensor Flag Clear Register

OTS0OTFCR is an 8-bit write-only register to control starting and terminating the temperature sensor. This register is always read as 0.

Access: This register can be written in 8-bit units.

Address: <OTSn_base> + $00C_H$

**Value after reset**: $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SDERC | — | OTFC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | W | R | W |

**Table 11.10 Contents of OTS0OTFCR Register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When writing, write the value after reset. |
| 2 | SDERC | Diagnosis Error Clear<br>Writing 0: Does not clear the diagnosis error.<br>Writing 1: Clears the diagnosis error. |
| 1 | Reserved | When writing, write the value after reset. |
| 0 | OTFC | Temperature Measurement End Flag Clear<br>Writing 0: Does not clear the temperature measurement end flag.<br>Writing 1: Clears the temperature measurement end flag. |

### 11.4.5 OTS0OTFR — Temperature Sensor Flag Register

OTS0OTFR is an 8-bit read-only register to indicate temperature sensor flags. OTS0OTFR is initialized to $00_H$ by any reset.

Access: This register can be read in 8-bit units.

Address: <OTSn_base> + $010_H$

**Value after reset**: $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SDER | OTACT | OTF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 11.11 Contents of OTS0OTFR Register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is read. |
| 2 | SDER | Diagnosis Error<br>Setting condition<br>A diagnosis error occurred.<br>Clearing condition<br>SDERC = 1 |
| 1 | OTACT | Temperature Sensor Status<br>0: The temperature sensor is in the idle state.<br>1: The temperature sensor is operating. |
| 0 | OTF | Temperature Measurement End Flag Setting condition<br>A temperature measurement value is written to OTS0OTDR.<br>Clearing condition<br>OTS0OTDR is read when OTFC = 1. |

**CAUTION**

SDER is updated when a temperature measurement value is written to OTS0OTDR.

When SDER asserting condition and clear condition are conflicted, flag clear operation is given priority to.

### 11.4.6 OTS0OTSTR — Temperature Status Register

OTS0OTSTR is an 8-bit read-only register to indicate temperature sensor status. OTS0OTSTR is initialized to $00_H$ by any reset.

Access: This register can be read in 8-bit units.

Address: <OTSn_base> + $014_H$

**Value after reset**: $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TSTAT[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 11.12 Contents of OTS0OTSTR Register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is read. |
| 1, 0 | TSTAT[1:0] | Temperature Status<br>0 0: Ordinary temperature (value after reset)<br>0 1: High temperature B<br>1 0: High temperature A<br>1 1: Low temperature A<br>When temperature is measured, temperature status is determined by the temperature measurement value and the status is reflected in TSTAT[1:0]. **Table 11.13** shows status transitions of TSTAT. **Table 11.14** shows temperature alarm error and temperature rise/drop interrupt generating conditions. |

**CAUTION**

The TSTAT[1:0] bits are updated when a temperature measurement value is written to OTS0OTDR.

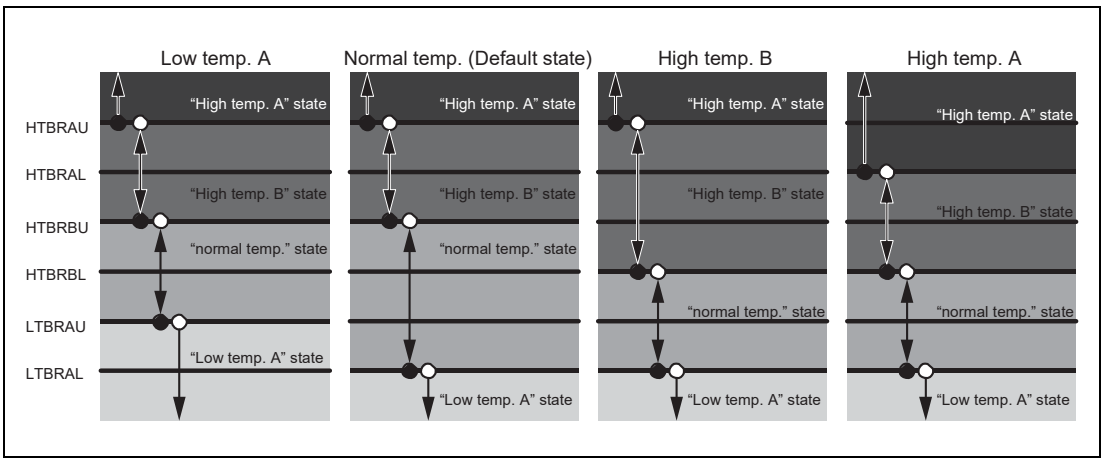



**Figure 11.2 Temperature status and transition states**

**Table 11.13 Temperature Status Transitions**

| | | After Status Transition | | | |
|---|---|---|---|---|---|
| | | **High temp. A** | **High temp. B** | **Ordinary temp.** | **Low temp. A** |
| **Before status transition** | **High temp. A** | # High A↓ | High A↓ and # High B↓ | High B↓ and # Low A↓ | Low A↓ |
| | **High temp. B** | High A↑ | # High A↑ and # High B↓ | High B↓ and # Low A↓ | Low A↓ |
| | **Ordinary temp.** | High A↑ | # High A↑ and High B↑ | # High B↑ and # Low A↓ | Low A↓ |
| | **Low temp. A** | High A↑ | # High A↑ and High B↑ | # High B↑ and Low A↑ | # Low A↑ |

This table shows TSTAT[1:0] status transitions before and after temperature measurement according to conditions of temperature measurement values.

[Legend]

High A ↑: High-temperature border AU register (OTS0HTBRAU) or more

High A ↓: Less than high-temperature border AL register (OTS0HTBRAL)

High B ↑: High-temperature border BU register (OTS0HTBRBU) or more

High B ↓: Less than high-temperature border BL register (OTS0HTBRBL)

Low A ↑: Low-temperature border AU register (OTS0LTBRAU) or more

Low A ↓: Less than low-temperature border AL register (OTS0LTBRAL)

#: Negation

**Table 11.14 Conditions for Generating Temperature Alarm Error and Temperature Rise/Drop Interrupts**

| | | After Status Transition | | | |
|---|---|---|---|---|---|
| | | **High temp. A** | **High temp. B** | **Ordinary temp.** | **Low temp. A** |
| **Before status transition** | **High temp. A** | **N/A** | **OTULI** | **OTULI** | **OTABE** |
| | **High temp. B** | **OTABE** | **N/A** | **OTULI** | **OTABE** |
| | **Ordinary temp.** | **OTABE** | **OTULI** | **N/A** | **OTABE** |
| | **Low temp. A** | **OTABE** | **OTULI** | **OTULI** | **N/A** |

[Legend]

OTABE: Occurrence of temperature alarm error

OTULI: Occurrence of temperature rise or drop interrupt

N/A: Neither OTABE nor OTULI occurs.

### 11.4.7 OTS0OTDR — Temperature Data Register

OTS0OTDR is a 16-bit read-only register that stores a temperature measurement value. OTS0OTDR is initialized to $0000_H$ by any reset.

Access: This register can be read in 16-bit units.

Address: <OTSn_base> + $018_H$

**Value after reset**: $0000_H$

Signed fixed-point format

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S*1 | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

↑

Decimal point position

Note 1. S: Sign bit (always 0)

As temperature guideline following table can be used.

**Table 11.15 temperature guideline**

| Register value | Temperature [°C] | Register value | Temperature [°C] |
|---|---|---|---|
| $1D95_H$ | –40 | $2B8C_H$ | 70 |
| $1ED9_H$ | –30 | $2CD1_H$ | 80 |
| $201E_H$ | –20 | $2E15_H$ | 90 |
| $2164_H$ | –10 | $2F59_H$ | 100 |
| $22A7_H$ | 0 | $309F_H$ | 110 |
| $23ED_H$ | 10 | $31E4_H$ | 120 |
| $2532_H$ | 20 | $3328_H$ | 130 |
| $2677_H$ | 30 | $346D_H$ | 140 |
| $27BA_H$ | 40 | $35B3_H$ | 150 |
| $2900_H$ | 50 | $36F9_H$ | 160 |
| $2A47_H$ | 60 | $383D_H$ | 170 |

The temperature formula is:

- Temperature [°C] = P × OTS0OTDR – Tm
  - P = 1008.3°C
  - Tm = 273.15°C
  - OTS0OTDR = sum of data times bit value

  Example Bit 14 => value of 1/2, bit 13 => value of 1/4,
  OTS0OTDR = $2800_H$ => value of 0.25 + 0.0625 => 1008.3 × 0.3125 – 273.15 = +42°C

**NOTE**

For an absolute temperature within the given accuracy (refer to **Section 35.7.4, Temperature Sensor Characteristics**), an offset has to be considered. The offset is given by trimming data in the register OTS0COEFFRA to OTS0COEFFRC. For correction calculation formula with coefficients, refer to **Section 11.4.11, OTS0COEFFRn — Coefficient n Register**.

## 11.4.8 OTS0HTBRmn — High-Temperature Border mn Register

OTS0HTBRmn is a 16-bit readable/writable register to specify the temperature border of high temperature m. OTS0HTBRmn is initialized to $0000_H$ by any reset.

**Access:** This register can be read/written in 16-bit units.

**Address:** OTS0HTBRAU: <OTSn_base> + $01C_H$, OTS0HTBRAL: <OTSn_base> + $020_H$
OTS0HTBRBU: <OTSn_base> + $024_H$, OTS0HTBRBL: <OTSn_base> + $028_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HTBmn | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 11.16 Contents of OTS0HTBRmn Register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | HTBmn | high-Temperature Border mn<br>These bits specify low-temperature border mn. Specify it so that the following condition is met.<br>OTS0HTBRAU > OTS0HTBRAL > OTS0HTBRBU > OTS0HTBRBL > OTS0LTBRAU > OTS0LTBRAL<br>The HTBmn format is the same as the OTS0OTDR format. HTBmn[15] is always 0.<br>For details, see **Section 11.4.6, OTS0OTSTR — Temperature Status Register** |

**CAUTION**

To prevent malfunction, set OTS0HTBRmn while the OTACT bit in OTS0OTFR is 0.

**NOTE**

m = A, B n = U, L

### 11.4.9 OTS0LTBRAn — Low-Temperature Border An Register

OTS0LTBRAn is a 16-bit readable/writable register to specify the temperature border of low temperature A. OTS0LTBRAn is initialized to $0000_H$ by any reset.

**Access:** This register can be read/written in 16-bit units.

**Address:** OTS0LTBRAU: <OTSn_base> + $02C_H$, OTS0LTBRAL: <OTSn_base> + $030_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LTBAn | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 11.17 Contents of OTS0LTBRAn Register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | LTBAn | Low-Temperature Border An<br>These bits specify low-temperature border An. Specify it so that the following condition is met.<br>OTS0HTBRAU > OTS0HTBRAL > OTS0HTBRBU > OTS0HTBRBL > OTS0LTBRAU > OTS0LTBRAL<br>The LTBAn format is the same as the OTS0OTDR format. LTBAn[15] is always 0.<br>For details, see **Section 11.4.6, OTS0OTSTR — Temperature Status Register** |

**CAUTION**

To prevent malfunction, set OTS0LTBRAn while the OTACT bit in OTS0OTFR is 0.

**NOTE**

n = U, L

### 11.4.10 OTS0TDLR — Temperature Difference Limiting Register

OTS0TDLR is a 16-bit readable/writable register to specify the temperature difference limit value for self-diagnosis. OTS0TDLR is initialized to $7FFF_H$ by any reset.

**Access:** This register can be read/written in 16-bit units.

**Address:** <OTSn_base> + $034_H$

**Value after reset:** $7FFF_H$

Signed fixed-point format

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TDL | | | | | | | | | | | | | | | |
| Value after reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

↑
Decimal point position

**Table 11.18 Contents of OTS0TDLR Register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | TDL | Temperature Difference Limit Value<br>These bits specify a value for limiting temperature difference between present temperature measurement value and previous temperature measurement value in continuous measurement mode. When the following condition is met, SDER (diagnosis error) is set to 1.<br>OTS0TDLR < \| present temperature measurement value – previous temperature measurement value \|<br>TDL[15] is always 0. |

**NOTE**

To prevent malfunction, set OTS0TDLR while the OTACT bit in OTS0OTFR is 0.

### 11.4.11 OTS0COEFFRn — Coefficient n Register

OTS0COEFFRn is a 16-bit read-only register to store coefficients for correction calculation.

Coefficients are stored in OTS0COEFFRn during shipping inspection. This data is expressed in two's complement form with sign.

**Access:** This register can be read in 16-bit units.

**Address:** OTS0COEFFRA: <OTSn_base> + $038_H$, OTS0COEFFRB: <OTSn_base> + $03C_H$, OTS0COEFFRC: <OTSn_base> + $040_H$

**Value after reset:** The value after reset will be stored at delivery inspection for each devices

Signed fixed-point format

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S*1 | | | | | | | | | | | | | | | |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

↑
Decimal point position

Note 1. S: Sign bit

- Reducing errors by temperature correction
  Three temperature correction coefficients are stored beforehand in coefficient registers A to C. Errors of temperature measurement values can be reduced by making a correction by a temperature correction calculation formula that uses temperature correction coefficients. However, temperature correction must be made by the CPU. By this the Temperature Border setting can be improved.

- Correction calculation formula: $T = AX^2 + BX + C$
  - T [K]: Temperature value after correction
  - X: OTS0OTDR
  - A to C: Coefficients A to C

**NOTE**

n = A to C

## 11.5 Operation

### 11.5.1 Functional description

- The temperature sensor sets six temperature threshold values in advance
- High-temperature border AU (OTS0HTBRAU) > High-temperature border AL (OTS0HTBRAL) > High-temperature border BU (OTS0HTBRBU) > High-temperature border BL (OTS0HTBRBL) > Low-temperature border AU (OTS0LTBRAU) > Low-temperature border AL (OTS0LTBRAL).
  Furthermore, the temperature sensor has four temperature states
  high temperature A, high temperature B, ordinary temperature, and low temperature A
  These temperature states are updated at each temperature measurement according to six temperature threshold values. The following describes conditions for generating a temperature alarm error and a temperature rise/drop interrupt.
- A temperature alarm error (OTABE) is output at a transition from a state other than high temperature A to high temperature A or at a transition from a state other than low temperature A to low temperature A.
- A temperature rise/drop interrupt (OTULI) is output at a transition from high temperature A to high temperature B or ordinary temperature or at a transition from high temperature B to ordinary temperature, that is, when temperature drops.
- A temperature rise/drop interrupt (OTULI) is output at a transition from low temperature A to high temperature B or ordinary temperature or at a transition from ordinary temperature to high temperature B, that is, when temperature rises.

### 11.5.2 Examples of Temperature Measurement Operation

The following figure shows temperature measurement operation in continuous measurement mode. There is no internal stabilization time in the second and subsequent temperature measurements.

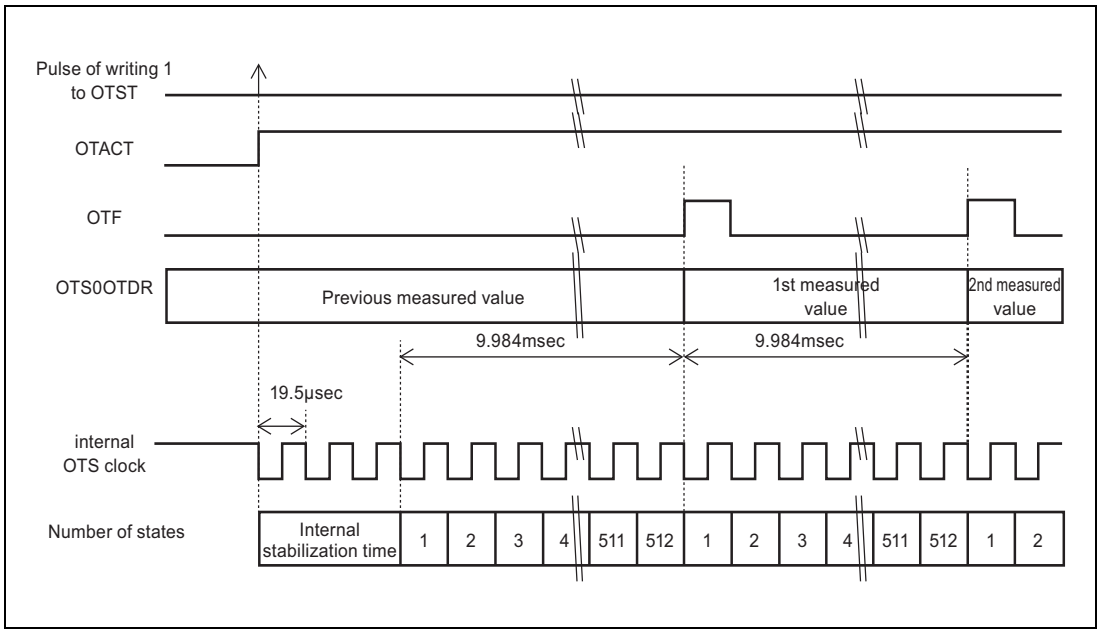



**Figure 11.3 Example of Temperature Measurement Operation (Continuous Measurement Mode)**

### 11.5.3 Temperature Measurement End Interrupt Request

The temperature sensor can generate a temperature measurement end interrupt request (OTI) to be sent to the INTC. At this time, the OTF bit in OTS0OTFR is set.

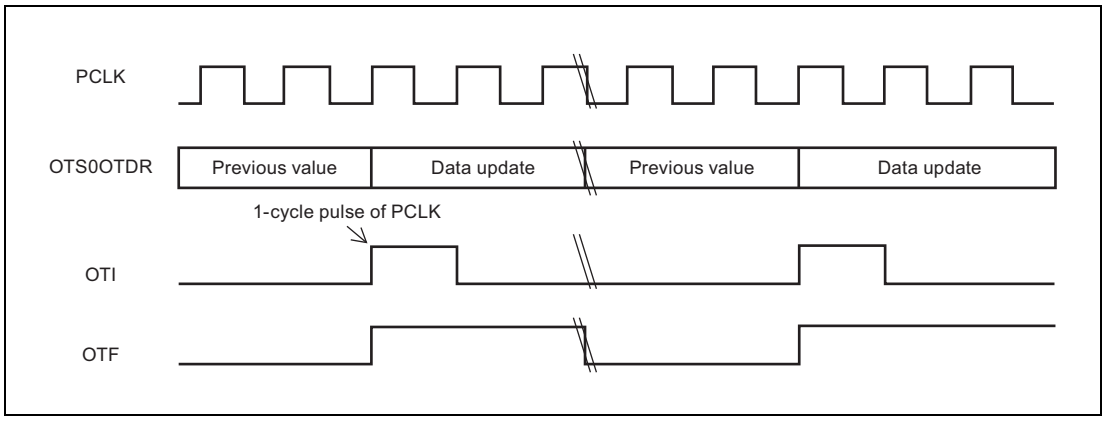



**Figure 11.4 Example of Occurrence of a Temperature Measurement End Interrupt**

When OTF asserting condition and clear condition are conflicted, flag clear operation is given priority to. But, when a result of measurement was renewed in the period until processing is suspended (within OTS 2 clock cycle) after the OTEND bit is set to 1, Temperature Sensor make OTI and OTF occur at the result of measurement renewal timing.

### 11.5.4 Temperature Alarm Error and Temperature Rise /Drop Interrupt and Temperature Sensor Error Interrupt Requests

The temperature sensor can generate a temperature rise or drop interrupt request(OTULI) and a temperature sensor error interrupt request (OTE) to be sent to the INTC. The temperature sensor can also generate a temperature alarm error (OTABE) to be sent to the error control module (ECM). Setting OTULIE in OTS0OTCR to 1 enables OTULI. Setting OTULIE to 0 disables OTULI. Setting OTEE in OTS0OTCR to 1 enables INTOTS0OTE. Setting OTEE to 0 disables INTOTS0OTE. Furthermore, setting OTABEE in OTS0OTCR to 1 enables OTABE. Setting OTABEE to 0 disables OTABE.

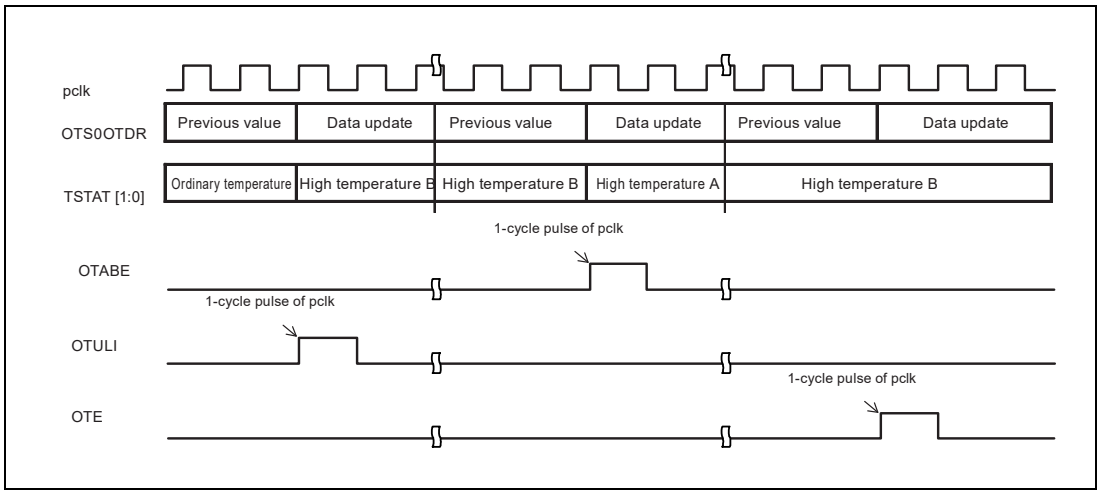



**Figure 11.5 Example of Occurrence of a Temperature Alarm Error and a Temperature Rise/Drop Interrupt and a Temperature Sensor Error Interrupt**

When Temperature Sensor do not measure temperature (OTFR.OTACT = 0), the initial value of temperature condition is "room temperature". After the OTSTCR.OTST bit is set to 1, and detecting the state except the room temperature, OTULI or OTABE are occurred (Detecting high-temperature B : OTULI, Detecting high-temperature A or low-temperature A : OTABE)

# Section 12 Clock Controller

The clock controller supplies clock pulses inside the chip and external devices. The clock controller consists of a main oscillator circuit (Main OSC), an internal oscillator, a Phase Locked Loop circuit (PLL), clock dividers and clock selectors. In RH850/P1x-C, the PLL uses the Main OSC as a reference clock and the Main OSC is mandatory for these devices.

## 12.1 Features

- Incorporates crystal resonation circuit (Main OSC), which is used as a reference clock of the PLL
- Incorporates internal oscillator, which is used until the PLL becomes stable, used as a backup clock during clock start up, and used for safety purpose.
- Incorporates PLL circuit to generate high speed internal clocks by multiplying the Main OSC input.
- Generates clock pulses used inside the chip from internal oscillator, main oscillator and PLL.
- Software configurable dividers for system clock tree (divider 0), peripheral clock tree (divider 1), which enable flexible clock gear function.
- Software configurable external clock output (divider 2 and 3).
- Adjustable PLL frequency by stored parameters in user accessible FLASH area.
- Halts clock supply to each module in Module standby mode.

### 12.1.1 External Input / Output pins

**Table 12.1** shows the pins related to the clock controller.

Table 12.1 Pins Related to the Clock Controller

| Pin function name | Direction | Function |
|---|---|---|
| X1 | Input | Main OSC crystal resonator/external clock input |
| X2 | Output | Main OSC crystal resonator |
| EXTCLK0O | Output | Clock controller output (divider 2) |
| EXTCLK1O | Output | Clock controller output (divider 3) |

## 12.2 Overview

### 12.2.1 Type of clocks

**Table 12.2** shows the list of clocks, **Table 12.3** shows the operation clocks of each functional module

**Table 12.2 List of clocks**

| Clock name | Symbol | Clock frequency (MHz) | | | | Remarks |
|---|---|---|---|---|---|---|
| CPU clock | CLK_CPU | (CPU) | 240 | 160 | 120 | Divider 0<br>(maximum frequencies configured by FLASH option)<br>Clock source: PLL output or CLK_IOSC |
| | | (GRAM) | 120 (1wait) | 80 (1wait) | 60 (1wait) | |
| High speed system clock | CLK_HSB | | 80 | 80 | 60 | |
| Low speed system clock | CLK_LSB | | 40 | 40 | 30 | |
| CSIH peripheral clock | CLKP_C | | 160 | 160 | 160 | Divider 1<br>Clock source: PLL output or CLK_IOSC |
| High speed peripheral clock1 | CLKP_H1 | | 80 | 80 | 80 | |
| High speed peripheral clock2 | CLKP_H2 | | 80 | 80 | 40 | |
| Low speed peripheral clock | CLKP_L | | 40 | 40 | 40 | |
| Backup clock | CLK_IOSC | | 8 | 8 | 8 | Internal oscillator clock *1/2 |
| SWDTA counter clock | SWDTACLKI | | 640kHz / 2kHz | 640kHz / 2kHz | 640kHz / 2kHz | 1/250 or (1/250 * 1/320) of 160MHz clock generated by the PLL (Configured by FLASH option) |
| WDTA counter clock | WDTACLKI | | 8 / 0.25 | 8 / 0.25 | 8 / 0.25 | 1/1 or 1/32 of CLK_IOSC (Configured by FLASH option) |
| External clock out 0 | EXTCLK0O | | 1/1 to 1/1023 of CLK_CPU, CLK_LSB, CLK_IOSC or Main OSC | | | Divider 2<br>Maximum output frequency: 20MHz |
| External clock out 1 | EXTCLK1O | | 1/1 to 1/1023 of CLK_CPU, CLK_LSB, CLK_IOSC or Main OSC | | | Divider 3<br>Maximum output frequency: 20MHz |

**Table 12.3 Clocks and functional modules**

| Clock name | Functional module name |
|---|---|
| CPU clock | PE1, PE2 (Master, Checker, LRAM, etc.), GRAM (1wait), Flash (1 or 2 wait) |
| High speed system clock | ICUMC, GTM, INTC, DMAC, DTS, ECM, DCRB, PIC, STM, MEMC, ETNA |
| Low speed system clock | SENT, ADCA, Clock controller |
| CSIH peripheral clock | CSIH |
| High speed peripheral clock1 | FLX, SENT |
| High speed peripheral clock2 | HS-USRT, MCAN/ M_TTCAN |
| Low speed peripheral clock | RLIN, OTS |
| Backup clock | CLMA |
| SWDT counter clock | SWDT |
| WDTA counter clock | WDTA |

**Note:** Backup clock is supplied until the PLL oscillation becomes stable.

### 12.2.2 Block Diagram

**Figure 12.1** shows the block diagram of clocks.

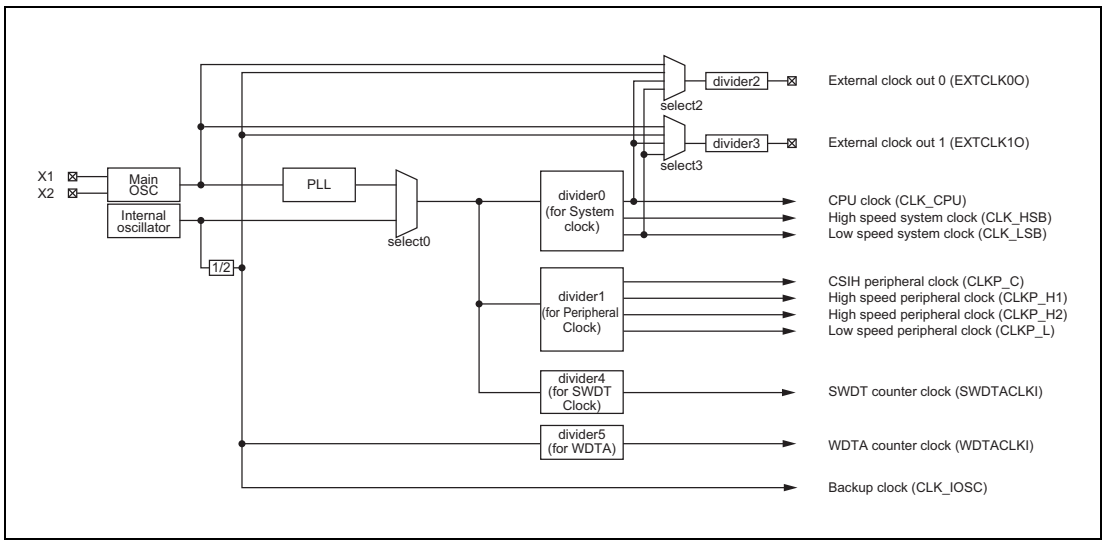



**Figure 12.1 Clock diagram (user's perspective)**

### 12.2.3 Clock sources

- Internal oscillator: 16MHz
- Main oscillator (Main OSC, crystal resonator or external clock): 16/20/24 MHz
- Phase Locked Loop (PLL)

### 12.2.4 PLL for the System Clock

The device may provide one dedicated PLL for the System Clock that is supplied to the CPU and related macros.

The Main Oscillator is used as a reference clock.

This PLL allows to select the operating frequencies of 120, 160, 240 MHz for the CPU and 80 MHz for the peripherals based on the preferred crystal input frequency of 16, 20, or 24MHz.

### 12.2.5 PLL Configuration

In RH850/P1x-C, the PLL macro is used. It provides various options for selecting the output frequency depending on flexible input clock requirements.

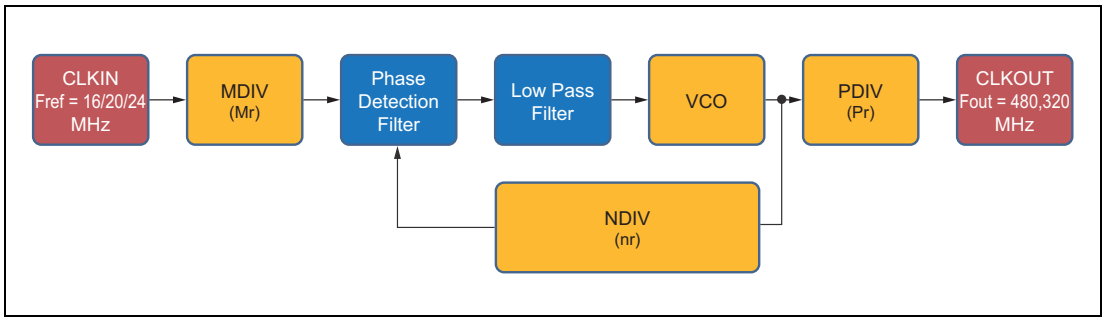



**Figure 12.2 PLL diagram**

The supported configurations are given in **Table 12.4, PLL and Clock divider setting.** Other input and output frequencies are not supported.

**Table 12.4 PLL and Clock divider setting**

| No | CLKIN [MHz] | MDIV (OPBT1. PLL0MDIV) | NDIV (OPBT1. PLL0NDIV) | PDIV (OPBT1. PLL0PDIV) | CLKOUT [MHz] | OPBT1. PLL0FREQ | CLK_CPU [MHz] | Normal operation mode | Serial Flash Programing mode |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 16 | 2 (001) | 60 (0x3b) | 1 (000) | 480 | $0_H$ | 120 (480/4) | √ | √ |
| 2 | 16 | 2 (001) | 60 (0x3b) | 1 (000) | 480 | $2_H$ | 240 (480/2) | √ | √ |
| 3 | 20 | 2 (001) | 48 (0x2f) | 1 (000) | 480 | $0_H$ | 120 (480/4) | √ | √ |
| 4 | 20 | 2 (001) | 48 (0x2f) | 1 (000) | 480 | $2_H$ | 240 (480/2) | √ | √ |
| 5 | 24 | 3 (010) | 60 (0x3b) | 1 (000) | 480 | $0_H$ | 120 (480/4) | √ | √ |
| 6 | 24 | 3 (010) | 60 (0x3b) | 1 (000) | 480 | $2_H$ | 240 (480/2) | √ | √ |
| 7 | 16 | 2 (001) | 40 (0x27) | 1 (000) | 320 | $1_H$ | 160 (320/2) | √ | √ |
| 8 | 20 | 2 (001) | 32 (0x1f) | 1 (000) | 320 | $1_H$ | 160 (320/2) | √ | √ |
| 9 | 24 | 3 (010) | 40 (0x27) | 1 (000) | 320 | $1_H$ | 160 (320/2) | √ | √ |
| 10*2 | 16 | 2 (001)*1 | 40 (0x27) *1 | 1 (000)*1 | 320 | $2_H$*1 | 160 (320/2) | — | √ |
| 11*2 | 20 | 2 (001)*1 | 40 (0x27) *1 | 1 (000)*1 | 400 | $2_H$*1 | 200 (400/2) | — | √ |
| 12*2 | 24 | 2 (001)*1 | 40 (0x27) *1 | 1 (000)*1 | 480 | $2_H$*1 | 240 (480/2) | — | √ |

Note 1. Initial setting of Option Byte 1 (OPBT1).

Note 2. No.10,11 and 12 are available only in the Serial programming mode. Set configuration No 1-9 before using the chip in normal operation mode.

### 12.2.6 Main Oscillator

The Main Oscillator clock is used as the reference clock for the PLLs.

- There are two ways of supplying the clock signal to the main oscillator. One is to connecting a crystal resonator, the another is to inputting an external clock signal. It can be selected by EXCLKIN FLASH options in **Section 32, Flash Memory**.
- A crystal resonator can be connected to X1 and X2. In this case, it is adjusted to the frequency of the crystal resonator by AMPSEL FLASH options in **Section 32, Flash Memory**. External Reset must not be released, before it is stabilized.

### 12.2.7 Internal Oscillator

The Internal Oscillator is used as Backup Clock during clock start up.

- It generates a clock, which runs at a frequency of 16 MHz It starts operation during power up and can not be stopped.

## 12.3 Register Description

### 12.3.1 Writing to protected registers

Registers can be protected from inadvertent write access due to erroneous program execution, etc. by configuration of the Slave Guards. For details, see **Section 28, Functional Safety**.

### 12.3.2 Register Overview

The Clock Controller is controlled and operated by the following registers:

**Table 12.5 List of registers**

| Address | Register Name | Description | Access Width | Access Protection | | |
|---|---|---|---|---|---|---|
| | | | | Value after reset | PBG | other |
| FFF8 8800$_H$ | CLKD0DIV | Clock divider 0 divisor register | 32 | 0000 0001$_H$ | PBG4#2.SC5 | — |
| FFF8 8804$_H$ | CLKD0STAT | Clock divider 0 status register | 32 | 0000 0003$_H$ | PBG4#2.SC5 | — |
| FFF8 8808$_H$ | CLKD1DIV | Clock divider 1 divisor register | 32 | 0000 0001$_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 880C$_H$ | CLKD1STAT | Clock divider 1 status register | 32 | 0000 0003$_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 8810$_H$ | CLKD2DIV | Clock divider 2 divisor register | 32 | 0000 0001$_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 8814$_H$ | CLKD2STAT | Clock divider 2 status register | 32 | 0000 0003$_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 8818$_H$ | CLKD3DIV | Clock divider 3 divisor register | 32 | 0000 0001$_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 881C$_H$ | CLKD3STAT | Clock divider 3 status register | 32 | 0000 0003$_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 9000$_H$ | CKSC0C | Clock selector 0 control register | 32 | 0000 0001$_H$ | PBG4#2.SC5 | — |
| FFF8 9008$_H$ | CKSC0S | Clock selector 0 status register | 32 | 0000 0001$_H$ | PBG4#2.SC5 | — |
| FFF8 9080$_H$ | CKSC2C | Clock selector 2 control register | 32 | 0000 0004$_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 9088$_H$ | CKSC2S | Clock selector 2 status register | 32 | 0000 0004$_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 90C0$_H$ | CKSC3C | Clock selector 3 control register | 32 | 0000 0004$_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 90C8$_H$ | CKSC3S | Clock selector 3 status register | 32 | 0000 0004$_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 4000$_H$ | CKSC4C | Clock selector 4 control register | 32 | 0000 0008$_H$ | PBG4#0. PG4-SC2 | SWDOPRUN |
| FFF8 4008$_H$ | CKSC4S | Clock selector 4 status register | 32 | 0000 0008$_H$ | PBG4#0. PG4-SC2 | — |

Table 12.6 Register reset conditions

| Register Name | Reset Condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset1 | System Reset2 | Application Reset | Limited Reset |
| CLKD0DIV | √ | √ | — | — | — |
| CLKD0STAT | √ | √ | — | — | — |
| CLKD1DIV | √ | √ | — | — | — |
| CLKD1STAT | √ | √ | — | — | — |
| CLKD2DIV | √ | √ | √ | √ | — |
| CLKD2STAT | √ | √ | √ | √ | — |
| CLKD3DIV | √ | √ | √ | √ | — |
| CLKD3STAT | √ | √ | √ | √ | — |
| CKSC0C | √ | √ | — | — | — |
| CKSC0S | √ | √ | — | — | — |
| CKSC2C | √ | √ | √ | √ | — |
| CKSC2S | √ | √ | √ | √ | — |
| CKSC3C | √ | √ | √ | √ | — |
| CKSC3S | √ | √ | √ | √ | — |
| CKSC4C | √ | √ | √ | √ | — |
| CKSC4S | √ | √ | √ | √ | — |

### 12.3.3 CLKD0DIV — Clock divider 0 divisor register

This register defines the divider factor of the clock divider0 used for the system clock generation. This register must not be written with a new value while the CLKD0SYNC is de-asserted. This register should be protected by a slave guard.

This register is initialized by the Power-on reset and the System reset 1.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 8800$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CLKD0DIV | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 12.7 CLKD0DIV register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | CLKD0DIV | Divider factor<br>0: Setting prohibited<br>1..7: Change the divisors in the **Table 12.8** to **Table 12.11** |

**Note:** See "**Section 3.4.5, Clock Ratio Change**" of **Section 3 CPU System** and "**Section 32.15, Usage Notes**" of **Section 32 Flash Memory**.

**Table 12.8 System Clock divisor table 0 (CLK_CPU: 240MHz) OPBT1.PLL0FREQ = 2$_H$ (PLL = 480MHz, CLK_CPU = 240MHz, CLKP_C = 160MHz)**

| CLKD0DIV | CLK_CPU | CLK_HSB | CLK_LSB |
|---|---|---|---|
| 1$_H$ | 2 (240 MHz) | 6 (80 MHz) | 12 (40 MHz) |
| 2$_H$ | 3 (160 MHz) | 6 (80 MHz) | 12 (40 MHz) |
| 3$_H$ | 6 (80 MHz) | 6 (80 MHz) | 12 (40 MHz) |
| 4$_H$ | 6 (80 MHz) | 12 (40 MHz) | 12 (40 MHz) |
| 5$_H$ | 12 (40 MHz) | 12 (40 MHz) | 12 (40 MHz) |
| 6$_H$ | 24 (20 MHz) | 24 (20 MHz) | 24 (20 MHz) |
| 7$_H$ | 24 (20 MHz) | 24 (20 MHz) | 24 (20 MHz) |

**Table 12.9 System Clock divisor table 1 (CLK_CPU = 160MHz) OPBT1.PLL0FREQ = $1_H$ (PLL = 320 MHz, CLK_CPU = 160MHz, CLKP_C = 160MHz)**

| CLKD0DIV | CLK_CPU | CLK_HSB | CLK_LSB |
|---|---|---|---|
| $1_H$ | 2 (160 MHz) | 4 (80 MHz) | 8 (40 MHz) |
| $2_H$ | 2 (160 MHz) | 4 (80 MHz) | 8 (40 MHz) |
| $3_H$ | 4 (80 MHz) | 4 (80 MHz) | 8 (40 MHz) |
| $4_H$ | 4 (80 MHz) | 8 (40 MHz) | 8 (40 MHz) |
| $5_H$ | 8 (40 MHz) | 8 (40 MHz) | 8 (40 MHz) |
| $6_H$ | 16 (20 MHz) | 16 (20 MHz) | 16 (20 MHz) |
| $7_H$ | 16 (20 MHz) | 16 (20 MHz) | 16 (20 MHz) |

**Table 12.10 System Clock divisor table 2 (CLK_CPU = 120MHz) OPBT1.PLL0FREQ = $0_H$ (PLL = 480MHz, CLK_CPU = 120MHz, CLKP_C = 160MHz)**

| CLKD0DIV | CLK_CPU | CLK_HSB | CLK_LSB |
|---|---|---|---|
| $1_H$ | 4 (120 MHz) | 8 (60 MHz) | 16 (30 MHz) |
| $2_H$ | 4 (120 MHz) | 8 (60 MHz) | 16 (30 MHz) |
| $3_H$ | 4 (120 MHz) | 12 (40 MHz) | 12 (40 MHz) |
| $4_H$ | 8 (60 MHz) | 16 (30 MHz) | 16 (30 MHz) |
| $5_H$ | 12 (40 MHz) | 12 (40 MHz) | 12 (40 MHz) |
| $6_H$ | 24 (20 MHz) | 24 (20 MHz) | 24 (20 MHz) |
| $7_H$ | 24 (20 MHz) | 24 (20 MHz) | 24 (20 MHz) |

**Table 12.11 System Clock divisors and frequencies (Clock source: Backup clock)**

| CLKD0DIV | CLK_CPU | CLK_HSB | CLK_LSB |
|---|---|---|---|
| All values | 2 (8 MHz) | 2 (8 MHz) | 2 (8 MHz) |

### 12.3.4 CLKD0STAT — Clock divider 0 status register

This register indicates the status of the clock divider0 used for the system clock generation

This register is initialized by the Power-on reset and the System reset 1.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 8804$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLKD0 SYNC | CLKD0 CLKACT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.12 CLKD0STAT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. |
| 1 | CLKD0SYNC | Divider clock synchronized<br>0: Clock output does not correspond to the actual divisor setting in CLKD0DIV.<br>1: Clock output corresponds to the actual divisor setting in CLKD0DIV. |
| 0 | CLKD0CLKACT | Divider clock active<br>0: Clock inactive<br>1: Clock active |

### 12.3.5 CLKD1DIV — Clock divider 1 divisor register

This register defines the divider factor of the clock divider1 used for the peripheral clock generation. This register must not be written with a new value while the CLKD1SYNC is de-asserted. This register should be protected by a slave guard.

This register is initialized by the Power-on reset and the System reset 1.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 8808$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CLKD1DIV | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 12.13 CLKD1DIV register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | CLKD1DIV | Divider factor<br>0: Setting prohibited<br>1..7 Change the divisors in the **Table 12.14** to **Table 12.17** |

**Table 12.14 Peripheral clock divisors and frequencies (CLK_CPU = 240MHz@MAX (PLL: 480 MHz))**
**OPBT1.PLL0FREQ = 2$_H$ (PLL = 480MHz, CLK_CPU = 240MHz, CLKP_C = 160MHz)**

| CLKD1DIV | CLKP_C | CLKP_H1 | CLKP_H2 | CLKP_L |
|---|---|---|---|---|
| 1$_H$ | 3 (160 MHz) | 6 (80 MHz) | 6 (80 MHz) | 12 (40 MHz) |
| 2$_H$ | 6 (80 MHz) | 6 (80 MHz) | 6 (80 MHz) | 12 (40 MHz) |
| 3$_H$ | 6 (80 MHz) | 12 (40 MHz) | 12 (40 MHz) | 12 (40 MHz) |
| 4$_H$ | 12 (40 MHz) | 12 (40 MHz) | 12 (40 MHz) | 12 (40 MHz) |
| 5 $_H$ | 12 (40 MHz) | 12 (40 MHz) | 12 (40 MHz) | 24 (20 MHz) |
| 6$_H$ | 12 (40 MHz) | 24 (20 MHz) | 24 (20 MHz) | 24 (20 MHz) |
| 7$_H$ | 24 (20 MHz) | 24 (20 MHz) | 24 (20 MHz) | 24 (20 MHz) |

**Table 12.15 Peripheral Clock divisors and frequencies (CLK_CPU = 160MHz@MAX (PLL: 320MHz))**
**OPBT1.PLL0FREQ = $1_H$ (PLL = 320MHz, CLK_CPU = 160MHz, CLKP_C = 160MHz)**

| CLKD1DIV | CLKP_C | CLKP_H1 | CLKP_H2 | CLKP_L |
|---|---|---|---|---|
| $1_H$ | 2 (160 MHz) | 4 (80 MHz) | 4 (80 MHz) | 8 (40 MHz) |
| $2_H$ | 4 (80 MHz) | 4 (80 MHz) | 4 (80 MHz) | 8 (40 MHz) |
| $3_H$ | 4 (80 MHz) | 8 (40 MHz) | 8 (40 MHz) | 8 (40 MHz) |
| $4_H$ | 8 (40 MHz) | 8 (40 MHz) | 8 (40 MHz) | 8 (40 MHz) |
| $5_H$ | 8 (40 MHz) | 8 (40 MHz) | 8 (40 MHz) | 16 (20 MHz) |
| $6_H$ | 16 (20 MHz) | 16 (20 MHz) | 16 (20 MHz) | 16 (20 MHz) |
| $7_H$ | 16 (20 MHz) | 16 (20 MHz) | 16 (20 MHz) | 16 (20 MHz) |

**Table 12.16 Peripheral Clock divisors and frequencies (CLK_CPU = 120MHz@MAX (PLL: 480 MHz))**
**OPBT1.PLL0FREQ = $0_H$ (PLL = 480MHz, CLK_CPU = 120MHz, CLKP_C = 160MHz)**

| CLKD1DIV | CLKP_C | CLKP_H1 | CLKP_H2 | CLKP_L |
|---|---|---|---|---|
| $1_H$ | 3 (160 MHz) | 6 (80 MHz) | 12 (40 MHz) | 12 (40 MHz) |
| $2_H$ | 6 (80 MHz) | 6 (80 MHz) | 12 (40 MHz) | 12 (40 MHz) |
| $3_H$ | 6 (80 MHz) | 12 (40 MHz) | 12 (40 MHz) | 12 (40 MHz) |
| $4_H$ | 12 (40 MHz) | 12 (40 MHz) | 12 (40 MHz) | 12 (40 MHz) |
| $5_H$ | 12 (40 MHz) | 12 (40 MHz) | 12 (40 MHz) | 24 (20 MHz) |
| $6_H$ | 12 (40 MHz) | 24 (20 MHz) | 24 (20 MHz) | 24 (20 MHz) |
| $7_H$ | 24 (20 MHz) | 24 (20 MHz) | 24 (20 MHz) | 24 (20 MHz) |

**Table 12.17 Peripheral Clock divisors and frequencies (Clock source: Backup clock)**

| CLKD1DIV | CLKP_C | CLKP_H1 | CLKP_H2 | CLKP_L |
|---|---|---|---|---|
| All values | 2 (8 MHz) | 2 (8 MHz) | 2 (8 MHz) | 2 (8 MHz) |

### 12.3.6 CLKD1STAT — Clock divider 1 status register

This register indicates the status of the clock divider1 used for the peripheral clock generation.

This register is initialized by the Power-on reset and the System reset 1.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 880C$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | | CLKD1 SYNC | CLKD1 CLKACT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.18 CLKD1STAT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. |
| 1 | CLKD1SYNC | Divider clock synchronized<br>0: Clock output does not correspond to the actual divisor setting in CLKD1DIV<br>1: Clock output corresponds to the actual divisor setting in CLKD1DIV |
| 0 | CLKD1CLKACT | Divider clock active<br>0: Clock inactive<br>1: Clock active |

### 12.3.7 CLKD2DIV — Clock divider 2 divisor register

This register defines the divider factor of the clock divider2 used for the external clock 0. This register must not be written with a new value while the CLKD2SYNC is de-asserted. This register should be protected by a slave guard.

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 $8810_H$

**Value after reset:** 0000 $0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CLKD2DIV[9:0] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 12.19 CLKD2DIV register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 10 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 9 to 0 | CLKD2DIV | Divider factor<br>0: The external clock is stopped.<br>1..1023: The source for the external clock is divided by the CLKD2DIV. |

### 12.3.8 CLKD2STAT — Clock divider 2 status register

This register indicates the status of the clock divider2 used for the external clock 0.

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 8814$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLKD2SYNC | CLKD2CLKACT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.20 CLKD2STAT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. |
| 1 | CLKD2SYNC | Divider clock synchronized<br>0: Clock output does not correspond to the actual divisor setting in CLKD2DIV<br>1: Clock output corresponds to the actual divisor setting in CLKD2DIV |
| 0 | CLKD2CLKACT | Divider clock active<br>0: Clock inactive<br>1: Clock active |

### 12.3.9 CLKD3DIV — Clock divider 3 divisor register

This register defines the divider factor of the clock divider3 used for the external clock 1. This register must not be written with a new value while the CLKD3SYNC is de-asserted. This register should be protected by a slave guard.

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 8818$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CLKD3DIV[9:0] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 12.21 CLKD3DIV register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 10 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 9 to 0 | CLKD3DIV | Divider factor<br>0: The external clock is stopped.<br>1..1023: The source for the external clock is divided by the CLKD3DIV. |

### 12.3.10 CLKD3STAT — Clock divider 3 status register

This register indicates the status of the clock divider3 used for the external clock 1.

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 881C$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLKD3 SYNC | CLKD3 CLKACT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.22 CLKD3STAT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. |
| 1 | CLKD3SYNC | Divider clock synchronized<br>0: Clock output does not correspond to the actual divisor setting in CLKD3DIV<br>1: Clock output corresponds to the actual divisor setting in CLKD3DIV |
| 0 | CLKD3CLKACT | Divider clock active<br>0: Clock inactive<br>1: Clock active |

### 12.3.11 CKSC0C — Clock selector 0 control register

This register selects the clock which drives the system and peripheral clock trees. This register must not be written with a new value while the CKSC0S shows different ID. This register is protected by a slave guard.

This register is initialized by the Power-on reset and the System reset 1.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 $9000_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CKSC0 | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 12.23 CKSC0C register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | CKSC0 | Clock source control<br>Defines the ID of the clock source to be selected next.<br>ID = $1_H$: PLL<br>ID = $2_H$: Internal oscillator<br>Other than above: setting prohibited. |

**Note:** See "**Section 3.4.5, Clock Ratio Change**" of **Section 3 CPU System** and "**Section 32.15, Usage Notes**" of **Section 32 Flash Memory**.

### 12.3.12 CKSC0S — Clock selector 0 status register

This register indicates the status of the clock selector 0.

This register is initialized by the Power-on reset and the System reset 1.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 $9008_H$

**Value after reset:** 0000 $0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CLKACT0 | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.24 CKSC0S register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. |
| 2 to 0 | CLKACT0 | Clock source status<br>Indicates the ID of the selected clock source of clock selector 0. This bit field means the clock source with the ID is selected or its de-selection is pending.<br>ID = $1_H$: PLL<br>ID = $2_H$: Internal oscillator |

## 12.3.13 CKSC2C — Clock selector 2 control register

This register selects the clock which drives the external clock output 0.

This register is initialized by the Power-on reset, the System reset 1/2, and the Application reset.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 $9080_H$

**Value after reset:** 0000 $0004_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CKSC2 | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 12.25 CKSC2C register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | CKSC2 | Clock source control<br>Defines the ID of the clock source to be selected next.<br>ID = $3_H$: Main OSC<br>ID = $4_H$: CLK_LSB<br>ID = $5_H$: CLK_CPU<br>ID = $6_H$: Backup clock<br>Other than above: setting prohibited. |

**NOTE**

CKSC2 should not be modified when CLKD2DIV is set to 0H. Please refer **12.4.2** which shows the procedure to modify this register.

### 12.3.14 CKSC2S — Clock selector 2 status register

This register indicates the status of the clock selector 2.

This register is initialized by the Power-on reset, the System reset 1/2, and the Application reset.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 9088$_H$

**Value after reset:** 0000 0004$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CLKACT2 | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.26 CKSC2S register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. |
| 2 to 0 | CKSC2 | Clock source status<br>Indicates the ID of the selected clock source of clock selector 2. This bit field means the clock source with the ID is selected or its de-selection is pending.<br>ID = 3$_H$: Main OSC<br>ID = 4$_H$: CLK_LSB<br>ID = 5$_H$: CLK_CPU<br>ID = 6$_H$: Backup clock |

### 12.3.15 CKSC3C — Clock selector 3 control register

This register selects the clock which drives the external clock output 1.

This register is initialized by the Power-on reset, the System reset 1/2, and the Application reset.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 90C0$_H$

**Value after reset:** 0000 0004$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CKSC3 | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 12.27 CKSC3C register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | CKSC3 | Clock source control<br>Defines the ID of the clock source to be selected next.<br>ID = 3$_H$: Main OSC<br>ID = 4$_H$: CLK_LSB<br>ID = 5$_H$: CLK_CPU<br>ID = 6$_H$: Backup clock<br>Other than above: setting prohibited. |

**NOTE**

CKSC3 should not be modified when CLKD3DIV is set to 0H. Please refer **12.4.2** which shows the procedure to modify this register.

### 12.3.16 CKSC3S — Clock selector 3 status register

This register indicates the status of the clock selector 3.

This register is initialized by the Power-on reset, the System reset 1/2, and the Application reset.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 90C8$_H$

**Value after reset:** 0000 0004$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CLKACT3 | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.28 CKSC3S register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. |
| 2 to 0 | CKSC3 | Clock source status<br>Indicates the ID of the selected clock source of clock selector 3. This bit field means the clock source with the ID is selected or its de-selection is pending.<br>ID = 3$_H$: Main OSC<br>ID = 4$_H$: CLK_LSB<br>ID = 5$_H$: CLK_CPU<br>ID = 6$_H$: Backup clock |

## 12.3.17 CKSC4C — Clock selector 4 control register

This register selects the clock source of the SWDTACLKI.

This register is initialized by the Power-on reset, the System reset 1/2, and the Application reset.

**Access:** This register can be read/write in 32-bit units.

**Address:** FFF8 4000$_H$

**Value after reset:** 0000 0008$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CKSC4 | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W*1 | R/W*1 | R/W*1 | R/W*1 |

Note 1. Update available only one time if the SWDOPRUN is 1. If update is needed, it should have been done before the SWDTA start running.

**Table 12.29 CKSC4C register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 3 to 0 | CKSC4 | Clock source control<br>Defines the ID of the clock source to be selected next.<br>Writing CKSC4 with 0 and writing CKSC4 with values not defined below are ignored.<br><SWDOPSCKS = 1><br>ID = 8$_H$: 1/250 <= Default<br>ID = 9$_H$: 1/250 * 1/320<br><SWDOPSCKS = 0><br>ID = 8$_H$: 1/250 * 1/320 <= Default<br>ID = 9$_H$: 1/250<br><SWDOPRUN = 0><br>This register cannot be updated.<br><SWDOPRUN = 1><br>This register can be updated only once.<br>(The CKSC4C register can be written only once, i.e. writing CKSC4C register will disable any subsequent write to the CKSC4C register.) |

### 12.3.18 CKSC4S — Clock selector 4 status register

This register indicates the status of the clock selector 4 (SWDTACLKI).

This register is initialized by the Power-on reset, the System reset 1/2, and the Application reset.

**Access:** This register can be read in 32-bit units.

**Address:** FFF8 4008$_H$

**Value after reset:** 0000 0008$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CLKACT4 | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 12.30 CKSC4S register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | These bits are always read as 0. |
| 3 to 0 | CLKACT4 | Clock source control<br>Indicates the ID of the selected clock source of clock selector 4. This bit field means the clock source with the ID is selected or its de-selection is pending.<br><SWDOPSCKS = 1><br>ID = $8_H$: 1/250 <= Default<br>ID = $9_H$: 1/250 * 1/320<br><SWDOPSCKS = 0><br>ID = $8_H$: 1/250 * 1/320 <= Default<br>ID = $9_H$: 1/250 |

## 12.4 Operation

### 12.4.1 Clock Selection for Operating Frequency

The device provides software configurable dividers for the System and Peripheral Clock Trees (see **Figure 12.1, Clock Tree Overview**). Depending on the setting of these dividers, the operating of those trees can be limited to 1/n of the respective target frequency. These dividers are prepared to realize a clock gear function to decrease the current consumption. In the initial state, the dividers are set to the target frequency. Each clock frequency can be reduced by using CLKD0DIV and CLKD1DIV registers. For a register description, see **Section 12.3.3, CLKD0DIV — Clock divider 0 divisor register, Section 12.3.4, CLKD0STAT — Clock divider 0 status register, Section 12.3.5, CLKD1DIV — Clock divider 1 divisor register, Section 12.3.6, CLKD1STAT — Clock divider 1 status register.**

Clock gear function could be used in the following case only. Please continue to use the clock gear with the initial setting usually.

1. If there is a need to change the system clock source to the backup clock.
2. If there is a requirement to decrease the power consumption when the microcontroller operation is not needed temporally.

Notes on writing to CLKD0DIV and CLKD1DIV registers are following.

1. To prevent the voltage drop and voltage rise caused by sudden change of current consumption, clock frequency should be changed step by step with any time intervals.
2. Peripherals should be set to the module standby mode beforehand if clock frequency will be changed (It is necessary to keep the module standby mode while the clock gear setting is changed from the initial setting).
3. Follow the procedure for changing these register:

   a. Read CLKDxSTAT.CLKDxSYNC and verify that the value of CLKDxSTAT.CLKDxSYNC is 1.

   b. Write new division ratio of the divider to the CLKDxDIV.CLKDxDIV.

   c. Read CLKDxSTAT.CLKDxSYNC and repeat until the value of CLKDxSTAT.CLKDxSYNC becomes 1.

   (x = 0; 1)
4. There are some notes other than above which are described in other chapter. Please refer **Section 3.4.5, Clock Ratio Change** of **Section 3** and **Section 32.15, Usage Notes** of **Section 32**.

Follow the procedure for changing CKSC0C and CKSC4C:

1. Write new ID to CKSCiC
2. Read CKSCiC repeatedly until new ID is read from CKSCiC.CLKACTi.

   (i = 0, 4)

When the internal oscillator has been selected as a clock source by CKSC0C, divide factor of system clock dividers and peripheral clock dividers becomes divide by 2 regardless of the setting of CLKD0DIV and CLKD1DIV(refer to **Table 12.11** and **Table 12.17**). In this case, please note CSIH cannot be used because the frequency relation between CLK_HSB and CLKP_C is not suitable for CSIH.

### 12.4.2 External Clock Output Pins (ECLK0/1)

The device provides 2 continuous external clock outputs that can be used as clock supply for external circuits. The clock selection can be configured by software. For details see **Section 12.3.7, CLKD2DIV — Clock divider 2 divisor register, Section 12.3.8, CLKD2STAT — Clock divider 2 status register, Section 12.3.9, CLKD3DIV — Clock divider 3 divisor register, Section 12.3.10, CLKD3STAT — Clock divider 3 status register, Section 12.3.13, CKSC2C — Clock selector 2 control register, Section 12.3.14, CKSC2S — Clock selector 2 status register, and Section 12.3.15, CKSC3C — Clock selector 3 control register, Section 12.3.16, CKSC3S — Clock selector 3 status register**.

- Clock signals can be output from the EXTCLKnO (n = 0, 1) pins by the clock output function.
- Output clock frequencies can be divided by the division circuit with register settings.
- Main OSC, CLK_LSB, CLK_CPU and Backup clock can be selected with register settings.

CLKD2DIV and CLKD3DIV and CKSC2C and CKSC3C are registers that should be changed according to the next procedure.

Procedure to change CLKD2DIV and CLKD3DIV:

1. Write new value to CLKDiDIV
2. Read CLKDiDIV (confirming of register write)
3. Wait until CLKDiSTAT.CLKDiSYNC becomes 1
4. Insert min. 4 times dummy reads of CLKDiSTAT
5. Write the same value as used at procedure 1 to CLKDiDIV again
6. Read CLKDiDIV (confirming of register write)
7. Wait until CLKDiSTAT.CLKDiSYNC becomes 1

   (i = 2; 3)

Procedure to change CKSC2C and CKSC3C:

1. Write $0x00_H$ to CLKDiDIV
2. Read CLKDiDIV (confirming of register write)
3. Wait until CLKDiSTAT.CLKDiSYNC becomes 1
4. Insert min. 4 times dummy reads of CLKDiSTAT
5. Write $0x00_H$ to CLKDiDIV again
6. Read CLKDiDIV (confirming of register write)
7. Wait until CLKDiSTAT.CLKDiSYNC becomes 1
8. Write new ID to CKSCiC
9. Read CKSCiC repeatedly until new ID is read from CKSCiC.CLKACTi.

   (i = 2; 3)

## 12.5 Usage Notes

### 12.5.1 How to Connect a Crystal Resonator

**Figure 12.3** shows how to connect a crystal resonator. When the resonator recommended by the company is used (the detail information is separately provided), any external component such as a load capacitor and a dumping resistor is not necessarily required for oscillation.

As shown in **Figure 12.3**, do not cross any other signal lines over the signal lines to the X1 and X2 pins.

Induction may inhibit proper oscillation.

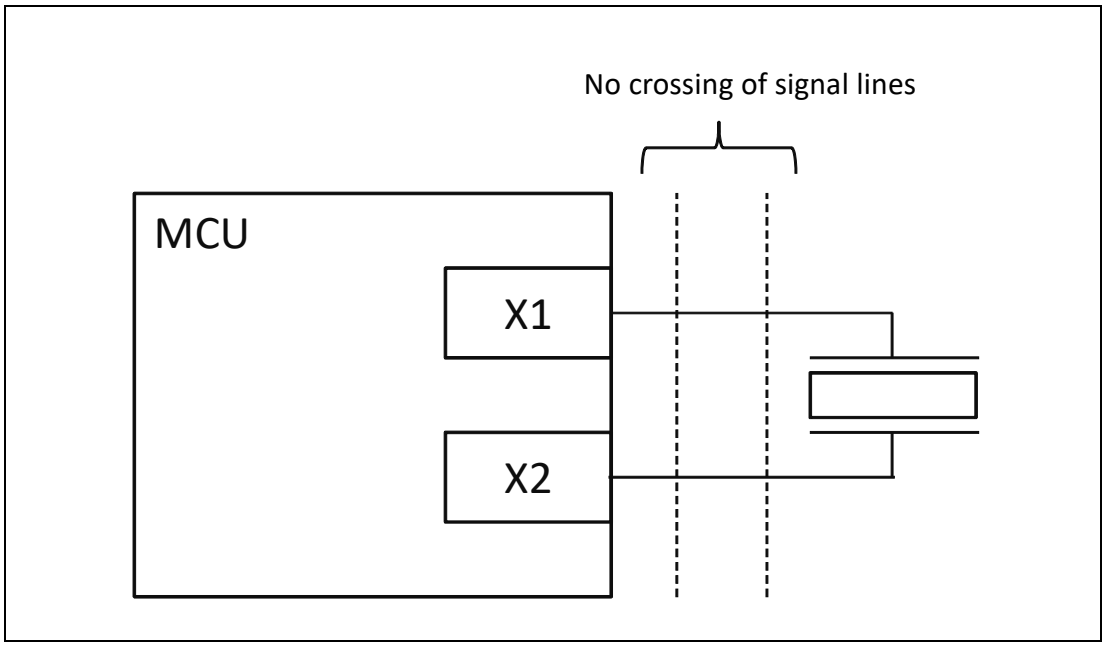



Figure 12.3 Connection Example of Crystal Resonator 1

## 12.6 Difference among P1M-C, P1H-C and P1H-CE

No differences in Clock controller but the targeted operating frequencies of P1H-C devices are 240 and 160MHz only.

# Section 13 Clock Monitor (CLMA)

The following sections describe clock monitor A (CLMA).

The first section describes the attributes specific to the RH850/P1x-C microcontrollers, including the number of channels, register base addresses, and input/output signal names.

The ensuing sections describe the functions relevant to all operations.

## 13.1 Features of RH850/P1x-C CLMA

### 13.1.1 Number of Channels

The RH850/P1x-C microcontrollers incorporate CLMA with the following number of channels.

**Table 13.1 Number of Channels**

| Product Name | P1M-C | P1H-C, P1H-CE |
|---|---|---|
| Number of channels | 5 | 6 |
| Name | CLMA0, CLMA1, CLMA2, CLMA3, CLMA5 | CLMA0, CLMA1, CLMA2, CLMA3, CLMA4, CLMA5 |

### 13.1.2 Register Base Addresses

The CLMA base addresses are listed in the following table.

The CLMA register addresses are expressed as offsets from the base addresses.

**Table 13.2 Register Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <CLMA0_base> | FFF8 $3100_H$ |
| <CLMA1_base> | FFF8 $3200_H$ |
| <CLMA2_base> | FFF8 $3300_H$ |
| <CLMA3_base> | FFF8 $3400_H$ |
| <CLMA4_base> | FFF8 $3500_H$ |
| <CLMA5_base> | FFF8 $6800_H$ |
| <CLMAT_base> | FFF8 $3000_H$ |

### 13.1.3 Clock Supply

The clocks monitored by CLMA and the CLMA sampling clocks are indicated below.

**Table 13.3 Clock Supply**

| Channel Name | Unit Clock Name | Clock Name |
|---|---|---|
| CLMA0 | CLMATMON (monitored clock) | Main OSC |
| | CLMATSMP (sampling clock) | CLK_IOSC / 2 |
| | PCLK (register access clock) | CLK_LSB |
| CLMA1 | CLMATMON (monitored clock) | CLKP_L ( to OTS0 ) |
| | CLMATSMP (sampling clock) | Main OSC / 4 |
| | PCLK (register access clock) | CLK_LSB |
| CLMA2 | CLMATMON (monitored clock) | WDTACLKI |
| | CLMATSMP (sampling clock) | Main OSC / 256 |
| | PCLK (register access clock) | CLK_LSB |
| CLMA3 | CLMATMON (monitored clock) | CLK_CPU (@PE1 checker core) / 2 |
| | CLMATSMP (sampling clock) | Man OCS / 4 |
| | PCLK (register access clock) | CLK_LSB |
| CLMA4 | CLMATMON (monitored clock) | CLK_CPU (@PE2 checker core) / 2 |
| | CLMATSMP (sampling clock) | Man OCS / 4 |
| | PCLK (register access clock) | CLK_LSB |
| CLMA5 | CLMATMON (monitored clock) | CLK_HSB (to ICUMC) |
| | CLMATSMP (sampling clock) | Man OCS / 4 |
| | PCLK (register access clock) | CLK_LSB |

**Note:** The PLL lock loss error status can be detected by CLMA1, CLMA3-5, because the PLL clock is monitored by theses clock monitors.

### 13.1.4 Reset Sources

CLMAn is initialized by the Power-on reset, the System reset 1/2, the Application reset, and register. CLMATEST.RESCLM can reset each CLMAn by special procedure. See **Section 13.5.3** and **Section 13.6.2** about CLMATEST.RESCLM.

**Table 13.4 CLMAn reset conditions**

| CLMA | Reset condition | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | CLMATEST.RESCLM | | | | | |
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset | CLMA0TESEN = 1 | CLMA1TESEN = 1 | CLMA2TESEN = 1 | CLMA3TESEN = 1 | CLMA4TESEN = 1 | CLMA5TESEN = 1 |
| CLMA0 | √ | √ | √ | √ | | √ | | | | | |
| CLMA1 | √ | √ | √ | √ | | | √ | | | | |
| CLMA2 | √ | √ | √ | √ | | | | √ | | | |
| CLMA3 | √ | √ | √ | √ | | | | | √ | | |
| CLMA4 | √ | √ | √ | √ | | | | | | √ | |
| CLMA5 | √ | √ | √ | √ | | | | | | | √ |

### 13.1.5 Internal Input/Output Signals

Table 13.5 Internal Input/Output Signals

| Clock Monitor | Output to |
|---|---|
| CLMA0 | ECM for PE1/PE2, Error factor #8 |
| CLMA1 | ECM for PE1/PE2, Error factor #12 |
| CLMA2 | ECM for PE1/PE2, Error factor #9 |
| CLMA3 | ECM for PE1, Error factor #10 |
| CLMA4 | ECM for PE2, Error factor #10 |
| CLMA5 | ECM for PE1/PE2, Error factor #11<br>ICUMC interrupt input INTCLMATI5 |

## 13.2 Overview

### 13.2.1 Functional Overview

Clock monitor CLMA detects frequency abnormalities in the monitored clock. It uses sampling clock CLMATSMP to monitor whether the frequency of input clock CLMATMON is within a specific range. Upon detection of an abnormal clock, it outputs an error notification to the ECM. In case of CLMA5 additionally an interrupt to the ICUMC.

The main components of the clock monitor are shown in **Figure 13.1**.

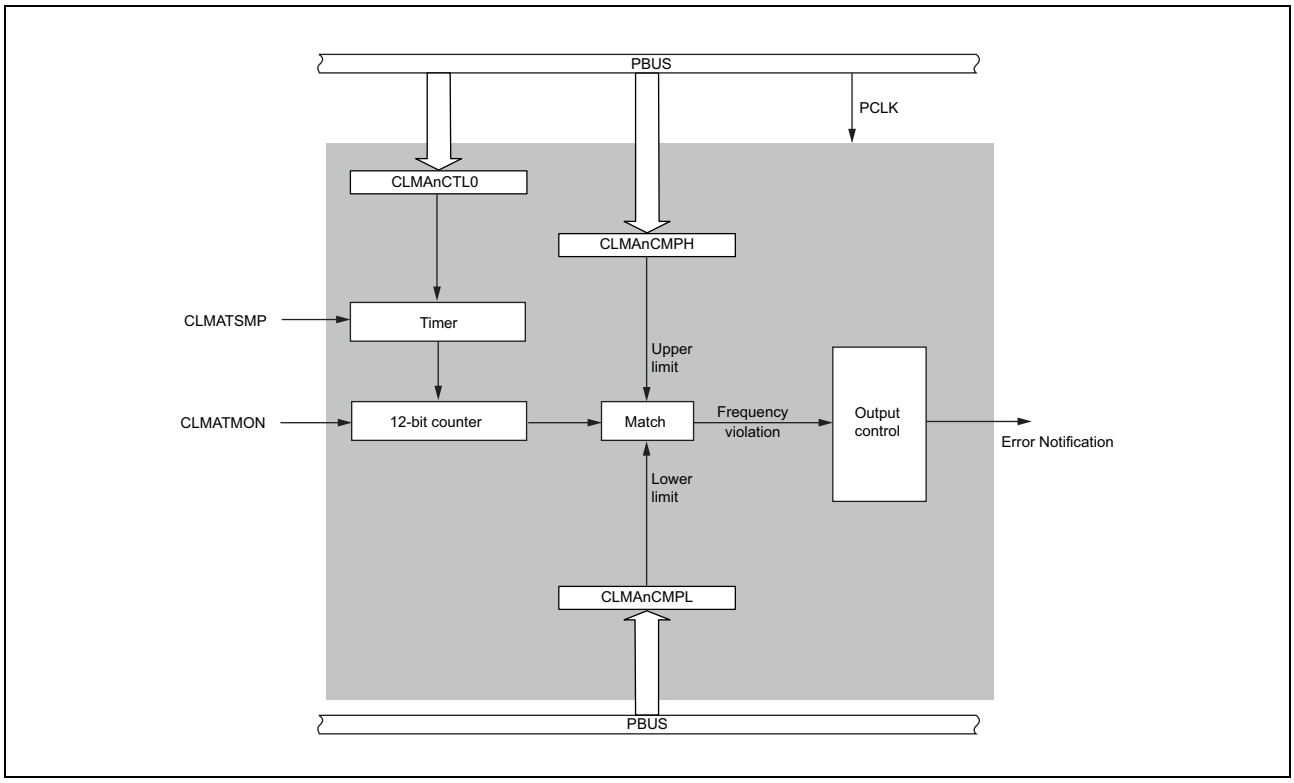



Figure 13.1 Block Diagram of the Clock Monitor A

## 13.3 Enabling CLMA

To enable CLMAn, CLMAnCTL0 which is a register to control CLMAn should be set $01_H$. The control register CLMAnCTL0 is a write protection register to enable CLMAn. See **Section 13.6.1** about procedure to enable CLMAn.

**NOTE**

- CLMAn can be only disabled by reset. Writing 0 to CLMAnCTL0 cannot disable CLMAn.
- The value after reset of CLMAnCTL0 is $00_H$. ($00_H$ means CLMAn is disabled)

## 13.4 Functions

The Clock Monitor CLMAn is used to verify whether the frequency of a clock (CLMATMON) is within the specified range.

### 13.4.1 Detection of Abnormal Clock Frequencies

**NOTE**

The clock monitor is not intended to detect a stopped clock (CLMATMON). There is a case that abnormal state is not detected when the monitor clock completely stops.

**Detection Method**

(1) CLMAn counts the rising edges of the monitored clock CLMATMON within 16 cycles of the sampling clock CLMATSMP and then compares the counter with the configured thresholds:
- CLMAnCMPL.CLMAnCMPL[11:0] defines the lower threshold.
- CLMAnCMPH.CLMAnCMPH[11:0] defines the upper threshold.

(2) When the monitored clock (CLMATMON) frequency is too low, the counter falls below CLMAnCMPL.CLMAnCMPL (Counter < CLMAnCMPL.CLMAnCMPL[11:0]).

(3) When the frequency of CLMATMON is too high, the counter exceeds CLMAnCMPH.CLMAnCMPH (CLMAnCMPH.CLMAnCMPH[11:0] ≤ Counter).

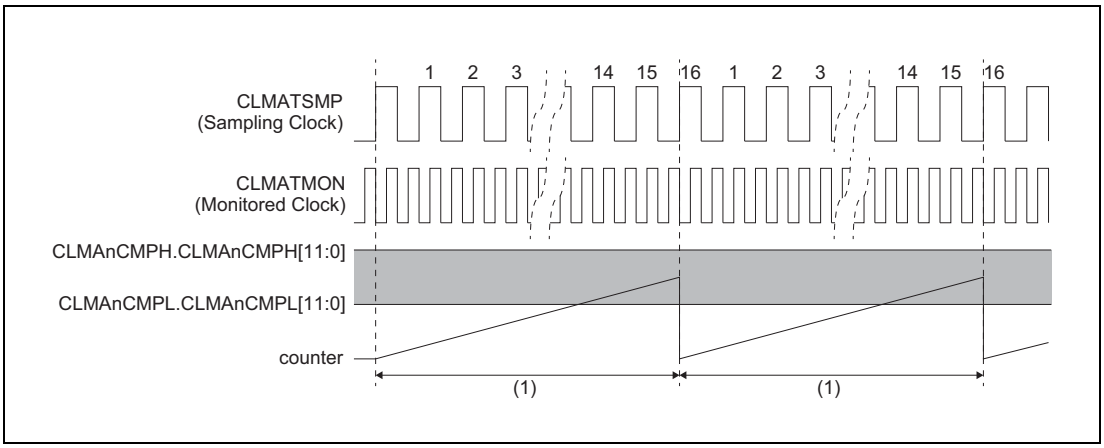



**Figure 13.2 Example: fCLMATMON is within the specified limit.**

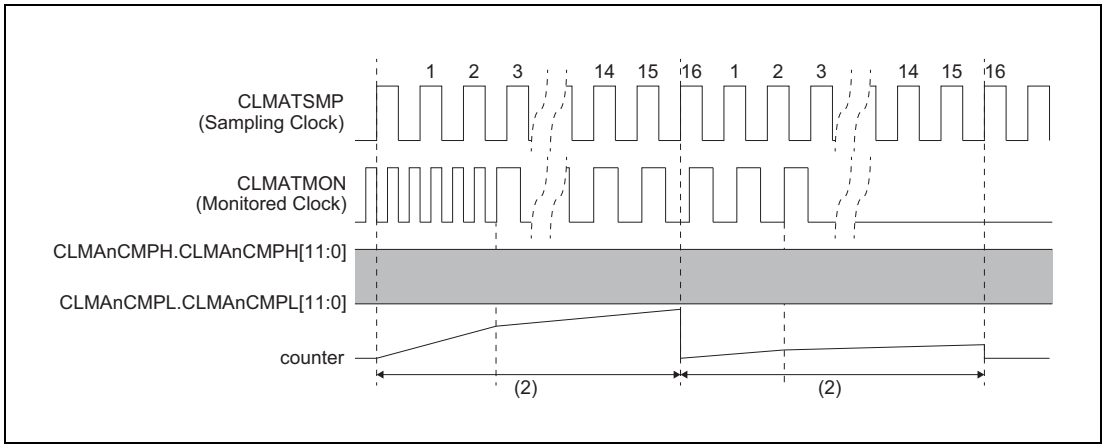



**Figure 13.3 Example: fCLMATMON is lower than the specified limit.**

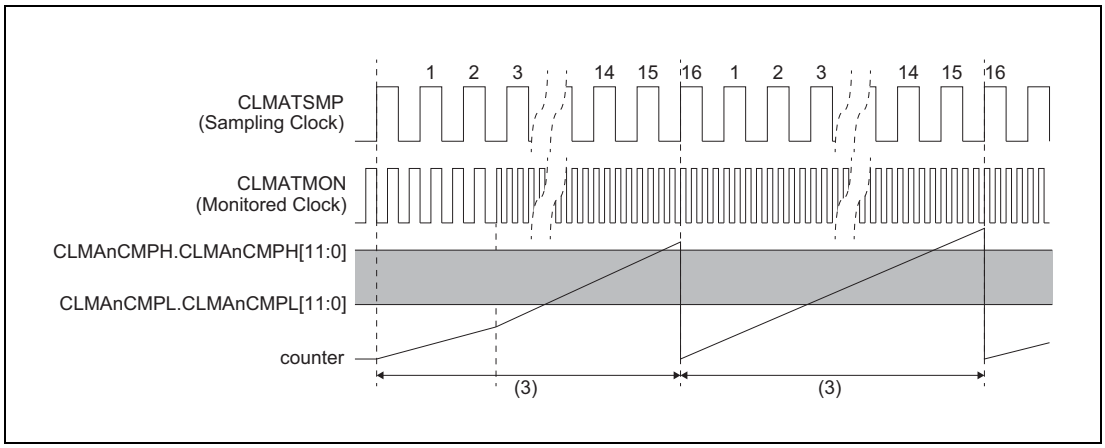



**Figure 13.4 Example: fCLMATMON is higher than the specified limit.**

**NOTE**

Even if the fCLMATMON exceeds or falls below the specified limits during a sample period, the counter might be within the valid range.

Abnormal fCLMATMON is detected after one sampling interval later.

### Calculation of thresholds

The compare registers CLMAnCMPL and CLMAnCMPH are configured with the minimum and maximum number of clock cycles of CLMATMON that are assumed to be valid within 16 cycles of the sampling clock CLMATSMP.

The expected number of clock cycles is denoted by N.

$$\frac{16}{f_{CLMATSMP}} = \frac{N}{f_{CLMATMON}}$$

$$N = \frac{f_{CLMATMON}}{f_{CLMATSMP}} \times 16$$

Considering the allowed frequency deviations of CLMATMON and CLMATSMP, the threshold values can be calculated by the following formulas:

$$\text{Lower threshold} = N_{min} = \frac{f_{CLMATMON(min)}}{f_{CLMATSMP(max)}} \times 16 - 1$$

$$\text{Upper threshold} = N_{max} = \frac{f_{CLMATMON(max)}}{f_{CLMATSMP(min)}} \times 16 + 1$$

### Example

For fCLMATSMP = 240 kHz (±8%) and fCLMATMON = 16 MHz (±5%) the recommended threshold values are the following:

$N_{min}$ = 15,200 / 259.2 × 16 - 1
= 937.27
CLMAnCMPL = 937 = $03A9_H$

$N_{max}$ = 16,800 / 220.8 × 16 + 1
= 1218.39
CLMAnCMPH = 1219 = $04C3_H$

### Minimum thresholds

The following restrictions must be taken into account:

- CLMAnCMPL ≥ $0001_H$
- CLMAnCMPH ≥ CLMAnCMPL + $0003_H$

## 13.4.2 Notification of Abnormal Clock Frequency

If $f_{CLMATMON}$ exceeds the upper threshold or falls below the lower threshold, this is indicated as follows:

- CLMAn sends an error notification to the ECM
- in case of CLMA5 additionally an interrupt to the ICUMC.

**NOTE**

The error notification is not negated until CLMAn is reset.

### 13.4.3 Self-Test

RH850/P1x-C implements a self-test function for the clock monitors.

It allows isolating the clock monitor from the system and executing functional test patterns via software. Each clock monitor can be tested individually. For performing these tests, the clock monitor inputs can be controlled by dedicated control registers, while its outputs can be observed in status registers.

Two registers are implemented for the self-test of the Clock Monitor, the Clock Monitor Test Register (CLMATEST) and the Clock Monitor Test Status Register (CLMATESTS). See the corresponding register descriptions for details.

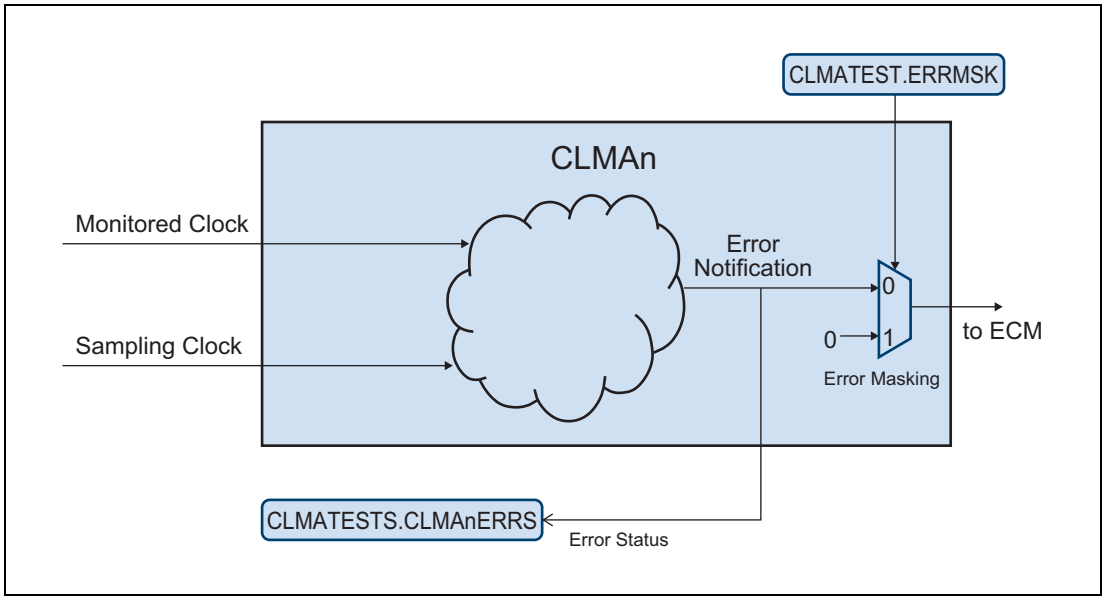



**Figure 13.5 Self-Test of the clock monitor**

# 13.5 Registers

## 13.5.1 Register Protection

Write protected registers are protected from inadvertent write access due to erroneous program execution, etc. Following Clock Monitor registers feature this special write protection:

- CLMAn control register 0 CLMAnCTL0

Other registers can be protected from inadvertent write access due to erroneous program execution, etc. by configuration of the P-Bus Guards. For details, see **Section 28, Functional Safety**.

And, CLMATEST.CLMA5TESEN is also protected from inadvertent write access by CLMA5CTL1.

## 13.5.2 List of Registers

The following table lists the CLMA registers.

<CLMAn_base> is defined in **Section 13.1.2, Register Base Addresses**.

**Table 13.6 List of Registers**

| Module name | Address | Register Name | Description | Access Width | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|---|
| | | | | | | PBG | other |
| CLMAn | <CLMAn_base> + $00_H$ | CLMAnCTL0 | CLMAn control register 0 | 8 | $00_H$ | PBG4#2.SC1 (n = 5) PBG4#0. PG4-SC3 (other) | CLMAnPCMD |
| CLMAn | <CLMAn_base> + $08_H$ | CLMAnCMPL | CLMAn comparison register L | 16 | $0001_H$ | PBG4#2.SC1 (n = 5) PBG4#0. PG4-SC3 (other) | accessible only when CLMAnCTL0.CLMAnCLME = 0 |
| CLMAn | <CLMAn_base> + $0C_H$ | CLMAnCMPH | CLMAn comparison register H | 16 | $03FF_H$ | PBG4#2.SC1 (n = 5) PBG4#0. PG4-SC3 (other) | accessible only when CLMAnCTL0.CLMAnCLME = 0 |
| CLMAn | <CLMAn_base> + $10_H$ | CLMAnPCMD | CLMAn protection command registrer | 8 | $00_H$ | PBG4#2.SC1 (n = 5) PBG4#0. PG4-SC3 (other) | |
| CLMAn | <CLMAn_base> + $14_H$ | CLMAnPS | CLMAn protection status register | 8 | $00_H$ | PBG4#2.SC1 (n = 5) PBG4#0. PG4-SC3 (other) | |
| CLMAn | <CLMAn_base> + $18_H$ | CLMAnEMU0 | CLMAn emuration register | 8 | $00_H$ | PBG4#2.SC1 (n = 5) PBG4#0. PG4-SC3 (other) | accessible only in break mode |
| CLMAC | <CLMAT_base> + $00_H$ | CLMATEST | Clock monitor test regsiter | 32 | $0000\ 0000_H$ | PBG4#0.PG4-SC3 | |
| CLMAC | <CLMAT_base> + $04_H$ | CLMATESTS | Clock monitor test status regsiter | 32 | $0000\ 0000_H$ | PBG4#0.PG4-SC3 | |
| CLMA5 | <CLMA5_base> + $04_H$ | CLMA5CTL1 | CLMA5 control register 1 | 8 | $00_H$ | PBG4#2SC1 | |

**NOTE**

CLMA4* registers cannot be written and is read as 0 in P1M-C.

### 13.5.3 Reset of Registers

Table 13.7 Register Reset Conditions

| Register Name | Reset condition | | | | |
|---|---|---|---|---|---|
| | **Power On** Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| All registgers | √ | √ | √ | √ | N/A |

CLMATEST.RESCLM can also reset each clock monitor (CLMA0/1/2/3/4/5). Then, before clock frequency is changed by register for clock during operations, it is necessary to go through special procedure and to retry setting parameters for CLMAn.

About special procedure, see **Section 13.6.2**.

Table 13.8 Individual Reset Conditions (1/2)

| Register Name | CLMATEST.RESCLM reset target | | | | | |
|---|---|---|---|---|---|---|
| | CLMA0TESEN = 1 | CLMA1TESEN = 1 | CLMA2TESEN = 1 | CLMA3TESEN = 1 | CLMA4TESEN = 1 | CLMA5TESEN = 1 |
| CLMA0CTL0 | √ | | | | | |
| CLMA0CMPL | √ | | | | | |
| CLMA0CMPH | √ | | | | | |
| CLMA0PCMD | √ | | | | | |
| CLMA0PS | √ | | | | | |
| CLMA0EMU0 | √ | | | | | |
| CLMA1CTL0 | | √ | | | | |
| CLMA1CMPL | | √ | | | | |
| CLMA1CMPH | | √ | | | | |
| CLMA1PCMD | | √ | | | | |
| CLMA1PS | | √ | | | | |
| CLMA1EMU0 | | √ | | | | |
| CLMA2CTL0 | | | √ | | | |
| CLMA2CMPL | | | √ | | | |
| CLMA2CMPH | | | √ | | | |
| CLMA2PCMD | | | √ | | | |
| CLMA2PS | | | √ | | | |
| CLMA2EMU0 | | | √ | | | |
| CLMA3CTL0 | | | | √ | | |
| CLMA3CMPL | | | | √ | | |
| CLMA3CMPH | | | | √ | | |
| CLMA3PCMD | | | | √ | | |
| CLMA3PS | | | | √ | | |
| CLMA3EMU0 | | | | √ | | |
| CLMA4CTL0 | | | | | √ | |
| CLMA4CMPL | | | | | √ | |

Table 13.8 Individual Reset Conditions (2/2)

| Register Name | CLMATEST.RESCLM reset target | | | | | |
|---|---|---|---|---|---|---|
| | CLMA0TESEN = 1 | CLMA1TESEN = 1 | CLMA2TESEN = 1 | CLMA3TESEN = 1 | CLMA4TESEN = 1 | CLMA5TESEN = 1 |
| CLMA4CMPH | | | | | √ | |
| CLMA4PCMD | | | | | √ | |
| CLMA4PS | | | | | √ | |
| CLMA4EMU0 | | | | | √ | |
| CLMA5CTL0 | | | | | | √ |
| CLMA5CMPL | | | | | | √ |
| CLMA5CMPH | | | | | | √ |
| CLMA5PCMD | | | | | | √ |
| CLMA5PS | | | | | | √ |
| CLMA5EMU0 | | | | | | √ |
| CLMATEST | | | | | | |
| CLMATESTS | | | | | | |
| CLMA5CTL1 | | | | | | √ |

### 13.5.4 CLMAnCTL0 — CLMAn Control Register 0

This register enables the clock monitor CLMAn.

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

CLMA4CTL0 (n = 4) cannot be written and is read as 0 in P1M-C.

#### Protection

Writing to this register is protected by a special sequence of instructions by using the protection command register CLMAnPCMD. See **Section 13.3, Enabling CLMA**.

**Access:** This register can be read/written in 8-bit units

**Address:** <CLMAn_base> + $00_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CLMAn CLME |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 13.9 CLMAnCTL0 Register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | These bits are always read as 0. Write value should always be 0. |
| 0 | CLMAnCLME | Enables or disables the clock monitor:<br>0: Disables CLMAn.<br>1: Enables CLMAn.<br>This bit can only be cleared by a reset |

### 13.5.5 CLMAnCMPL — CLMAn comparison register L

This register specifies the lower limit of monitored clock frequency.

Write access is permitted only when the CLMAn is disabled (CLMAnCTL0.CLMAnCLME = 0).

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

CLMA4CMPL (n = 4) cannot be written and is read as 0 in P1M-C.

**Access:** This register can be read/written in 16-bit units

**Address:** <CLMAn_base> + $08_H$

**Value after reset:** $0001_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | CLMAnCMPL[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 13.10 CLMAnCMPL Register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | These bits are always read as 0. Write value should always be 0. |
| 11 to 0 | CLMAnCMPL [11:0] | Specifies the lower threshold<br>The recommended value is (fCLMATMON(min) × 16) / fCLMATSMP(max) -1<br>The minimum value is $0001_H$ |

### 13.5.6 CLMAnCMPH — CLMAn comparison register H

This register specifies the upper limit of monitored clock frequency.

Write access is permitted only when the CLMAn is disabled (CLMAnCTL0.CLMAnCLME = 0).

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

CLMA4CMPH (n = 4) cannot be written and is read as 0 in P1M-C.

**Access:** This register can be read/written in 16-bit units

**Address:** <CLMAn_base> + $0C_H$

**Value after reset:** $03FF_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | CLMAnCMPH[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 13.11 CLMAnCMPH Register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | These bits are always read as 0. Write value should always be 0 |
| 11 to 0 | CLMAnCMPH [11:0] | Specifies the upper threshold<br>The recommended value is (fCLMATMON(max) × 16) / fCLMATSMP(min) +1<br>The minimum value is CLMAnCMPL + $0003_H$. |

### 13.5.7 CLMAnPCMD — CLMAn Protection command register

This register is a protection command register for the CLMAnCTL0 register against unintended writing. This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

CLMA4PCMD (n = 4) cannot be written in P1M-C.

**NOTE**

See **Section 13.6.1** for the procedure.

**Access:** This register can be written in 8-bit units

**Address:** <CLMAn_base> + $10_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CLMAnREG[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W |

**Table 13.12 CLMAnPCMD Register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | CLMAnREG [7:0] | Protection command that enables to write to clock monitor control protection cluster registers |

### 13.5.8 CLMAnPS — CLMAn Protection status register

This register is used to verify whether the write-protected register (CLMAnCTL0) has been successfully written or not.

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

CLMA4PS (n = 4) is read as 0 in P1M-C.

**Access:** This register can be read in 8-bit units

**Address:** <CLMAn_base> + $14_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CLMAn PRERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 13.13 CLMAnPS Register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | These bits are always read as 0. |
| 0 | CLMAnPRERR | Indicates whether the write-protected register (CLMAnCTL0) has been successfully written.<br>0: Write operation successful<br>1: Write operation failed |

### 13.5.9 CLMAnEMU0 — CLMAn Emulation Register 0

This register provides bits to emulate a frequency deviation error while the microcontroller is set in break mode during debugging. About break mode, See Section 31, On-chip Debug Unit (OCD).

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

CLMA4EMU0 (n = 4) cannot be written and is read as 0 in P1M-C.

**Access:** This register can be read/written in 8-bit units

**Address:** <CLMAn_base> + 18H

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CLMAn SLFST | CLMAn SLSLW |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 13.14 CLMAnEMU0 Register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | These bits are always read as 0. Write value should always be 0. |
| 1 | CLMAnSLFST | Specifies the higher value of fCLMATMON during emulation:<br>0: CLMATMON is assumed to be within the normal frequency range<br>1: CLMATMON is assumed to exceed the upper threshold. |
| 0 | CLMAnSLSLW | Specifies the lower value of fCLMATMON during emulation:<br>0: CLMATMON is assumed to be within the normal frequency range<br>1: CLMATMON is assumed to fall below the lower threshold. |

**CAUTION**

**It is prohibited to emulate a status of too low and too high frequency of CLMATMON at the same time. Thus CLMAnEMU0 must not be set to $03_H$.**

## 13.5.10 CLMATEST — Clock Monitor Test register

This register is used for the self-test of the clock monitors. Each Clock Monitor can be tested individually.

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

**Access:** This register can be read/written in 32-bit units

**Address:** <CLMAT_base> + $00_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CLMA5 TESEN | CLMA4 TESEN | CLMA3 TESEN | CLMA2 TESEN | CLMA1 TESEN | CLMA0 TESEN | ERR MSK | MONCL KMSK | RES CLM |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 13.15 CLMATEST Register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | These bits are always read as 0. Write value should always be 0. |
| 8 | CLMA5TESEN | CLMA5 self-test enable/disable<br>0: Test disabled.<br>1: Test enabled.<br>When CLMA5CTL1 is 0x1, this bit cannot be written. |
| 7 | CLMA4TESEN | CLMA4 self-test enable/disable<br>0: Test disabled.<br>1: Test enabled.<br>This bit can be read and written but the function is disabled in P1M-C. |
| 6 | CLMA3TESEN | CLMA3 self-test enable/disable<br>0: Test disabled.<br>1: Test enabled. |
| 5 | CLMA2TESEN | CLMA2 self-test enable/disable<br>0: Test disabled.<br>1: Test enabled. |
| 4 | CLMA1TESEN | CLMA1 self-test enable/disable<br>0: Test disabled.<br>1: Test enabled. |
| 3 | CLMA0TESEN | CLMA0 self-test enable/disable<br>0: Test disabled.<br>1: Test enabled. |
| 2 | ERRMSK*[1] | CLMA test error mask setting<br>Asserting this bit prevents the detected error from being forwarded into the device.<br>CLMA test reset signal mask setting<br>0: Error signal generation enabled<br>1: Error signal generation disabled (masked) |

**Table 13.15 CLMATEST Register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | MONCLKMSK *1 | Monitor clock mask setting<br>Asserting this bit disables the input of monitored clock to the associated Clock Monitor.<br>0: Monitored clock enabled<br>1: Monitored clock disabled (masked)<br>**CAUTION**<br>Do not use monitor clock masking function for self-test purpose |
| 0 | RESCLM*1 | CLMAn test reset signal control<br>Asserting this bit re-initializes the Clock Monitor in order to continue normal operation.<br>0: Reset signal for CLMAn is released.<br>1: Reset signal for CLMAn is asserted. |

Note 1. These bits are valid for CLMAn which is in self-test mode by setting “1” to CLMAnTESTEN.

**NOTE**

See **Section 13.6.2** for the procedure of the individual reset for CLMAn.

### 13.5.11 CLMATESTS — Clock Monitor Test Status register

This register is used for the self-test of the clock monitors. It monitors the error detection flags which are otherwise forwarded to the ECM module. Once error is detected, this register keeps the status until CLMAn is reset.

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

**Access:** This register can be read in 32-bit units

**Address:** <CLMAT_base> + $04_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CLMA5 ERRS | CLMA4 ERRS | CLMA3 ERRS | CLMA2 ERRS | CLMA1 ERRS | CLMA0 ERRS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 13.16 CLMATESTS Register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0 |
| 5 | CLMA5ERRS | CLMA5 error status<br>0: No error detected<br>1: Error detected |
| 4 | CLMA4ERRS | CLMA4 error status<br>0: No error detected<br>1: Error detected<br>This bit is read as 0 in P1M-C. |
| 3 | CLMA3ERRS | CLMA3 error status<br>0: No error detected<br>1: Error detected |
| 2 | CLMA2ERRS | CLMA2 error status<br>0: No error detected<br>1: Error detected |
| 1 | CLMA1ERRS | CLMA1 error status<br>0: No error detected<br>1: Error detected |
| 0 | CLMA0ERRS | CLMA0 error status<br>0: No error detected<br>1: Error detected |

### 13.5.12 CLMA5CTL1 — CLMA5 control register 1

This register is a write protection register for the CLMATEST.CLMA5TESEN.

This register is initialized by the Power-on reset, the System reset 1/2 and the Application reset.

**Access:** This register can be read/written in 8-bit units

**Address:** <CLMA5_base> + $04_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TESENWPRT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 13.17 CLMA5CTL1 Register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | These bits are always read as 0. Write value should always be 0. |
| 0 | TESENWPRT | Asserting the bit prevents the write access to the CLMATEST.CLMA5TESEN<br>0: enable to write CLMA5TESEN<br>1: disable to write CLMA5TESEN |

## 13.6 Operation

### 13.6.1 Procedures to enable CLMAn

To enable CLMAn, CLMAnCTL0 which is a register to control CLMAn should be set $01_H$.

Set $01_H$ to CLMAnCTL0 under the following instruction sequence.

**[Procedure]**

1. Write $A5_H$ in CLMAnPCMD.
2. Writing CLMAnCTL0 should be proceeded under the following sequence to enable CLMAn:
   - Write $01_H$ first.
   - Write the reversed value $FE_H$.
   - Write the target value $01_H$ again.
3. Read out the value of CLMAnCTL0.
   If the value of CLMAnCTL0 is $01_H$, CLMAn has been enabled.
   In other cases as shown below, the value of the CLMAnCTL0 write operation status register, CLMAnPS should be checked.
   - When CLMAnPS = $01_H$, the instruction sequence does not proceed correctly. Execute the sequence from step 1 again to enable CLMAn.
   - When CLMAnPS = $00_H$, write $00_H$ in CLMAnPCMD and then execute the sequence from step 1 again.

### 13.6.2 Procedures to Reset by CLMATEST.RESCLM

CLMATEST.RESCLM can also reset each clock monitor (CLMA0/1/2/3/4/5). Then, before clock frequency is changed by register for clock during operations, it is necessary to go through the procedure as below and to retry setting parameters for CLMAn.

**[Procedure]**

Example Case: Reset CLMA0

1. CLMATEST.CLMA0TESEN = 1 (write data : $0000_0008_H$)
2. CLMATEST.ERRMSK = 1 (write data : $0000_000C_H$)
3. CLMATEST.MONCLKMSK = 1 (write data : $0000_000E_H$)
4. CLMATEST.RESCLM = 1 (write data : $0000_000F_H$)
5. CLMATEST.RESCLM = 0 (write data : $0000_000E_H$)
6. CLMATEST.MONCLKMSK = 0 (write data : $0000_000C_H$)
7. CLMATEST.ERRMSK = 0 (write data : $0000_0008_H$)
8. CLMATEST.CLMA0TESEN = 0 (write data : $0000_0000_H$)

**NOTE**

The CLMA5CTL1.TESENWPRT should be set to 0 while setting the CLMATEST.CLMA5TESEN bit for CLMA5.

## 13.7 Difference among P1M-C, P1H-C and P1H-CE

P1M-C device doesn’t have the CLMA4. Other clock monitors are same in these devices.

# Section 14 Power Down Modes

## 14.1 Features

In order to reduce the current consumption, HALT mode and module standby are supported.

## 14.2 Power down Modes

### 14.2.1 HALT Mode

When the HALT instruction is executed, the CPU transits to HALT mode and stops instruction execution.

Each CPU (PE1, PE2 and ICUMC) can be controlled individually. **Table 14.1** shows the modules that participate in HALT Mode.

The CPU returns from this state by the occurrence of a reset input, interrupt, or exception.

**Table 14.1 Modules that participate in HALT mode**

| Module | Status during HALT Mode |
|---|---|
| PE1 | HALT/Run |
| PE2 | HALT/Run |
| ICUMC | HALT/Run |

### 14.2.2 Module standby

This function stops the clocks for peripheral macros to reduce the power consumption in accordance with register settings. **Table 14.2** shows the modules that participate in the module standby modes.

After reset is released, all peripherals except ICUMC enter module standby modes.

Example of the procedure of module standby mode is shown below.

#### Transition to Module standby mode

(1) Check the modules that participate in the module standby mode have completed the operation and are in idle state and no other module or external pin may activate the module.
For the details of the way to confirm the idle state of the modules, see the section related to each module.

(2) Check the Software Limited Reset Status Register of modules that participate in the module standby mode is 0. (SWLRESSx.SWLRESx_0 = 0)

(3) Stop all target clock domains that participate in the module standby mode.
(MSR_LMxx.MS_LMxx = 1)

(4) Read the value of MSRxxxx.MS_LMxx register and check the value is 1.

### Canceling Module standby mode

(1) Check the Software Limited Reset Status Register of modules that participate in the module standby mode is 0. (SWLRESSx.SWLRESx_0 = 0)

(2) Start all target clock domains that cancels module standby mode. (MSR_LMxx.MS_LMxx = 0)

(3) Read the value of MSR_xxxx.MS_LMxx register and check the value is 0.

(4) Execute the activating procedures of the modules which participate in the module standby mode. For the details, see the section related to each module.

**Table 14.2 Modules that participate in Module standby mode**

| Module standby register | Module |
|---|---|
| MSR_LM2 | ICUMC |
| MSR_LM3 | M-CAN/M-TTCAN |
| MSR_LM4 | FlexRay |
| MSR_LM5 | GTM |
| MSR_LM6 | Ethernet |
| MSR_LM7 | SENT |
| MSR_LM8 | HS-USRT |
| MSR_LM10 | CSIH |
| MSR_LM11 | RLIN3 |
| MSR_LM12 | ADCF |

## 14.3 Register Description

### 14.3.1 List of Registers

Table 14.3 Register Configuration

| Address | Register Name | Description | Access Width | Value after reset | PBG | other |
|---|---|---|---|---|---|---|
| FFF8 6210 | MSR_LM2 | Module standby Register for ICUMC | 32 | 0000 $0000_H$ | PBG4#2. SC1 | — |
| FFF8 1710 | MSR_LM3 | Module Standby Register for MCAN | 32 | 0000 $0001_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 1810 | MSR_LM4 | Module Standby Register for FlexRay | 32 | 0000 $0001_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 1910 | MSR_LM5 | Module Standby Register for GTM | 32 | 0000 $0001_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 1A10 | MSR_LM6 | Module Standby Register for Ethernet | 32 | 0000 $0001_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 1B10 | MSR_LM7 | Module Standby Register for SENT | 32 | 0000 $0001_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 1C10 | MSR_LM8 | Module Standby Register for HS-USRT | 32 | 0000 $0001_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 1E10 | MSR_LM10 | Module Standby Register for CSI-H | 32 | 0000 $0001_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 1F10 | MSR_LM11 | Module Standby Register for RLIN3 | 32 | 0000 $0001_H$ | PBG4#0. PG4-SC3 | — |
| FFF8 2010 | MSR_LM12 | Module Standby Register for AD | 32 | 0000 $0001_H$ | PBG4#0. PG4-SC3 | — |

Table 14.4 Register reset condition

| Register Name | Reset condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| MSR_LM2 | √ | √ | √ | √ | Limited Reset for ICUMC |
| MSR_LM3 | √ | √ | √ | √ | Limited Reset for MCAN |
| MSR_LM4 | √ | √ | √ | √ | Limited Reset for FlexRay |
| MSR_LM5 | √ | √ | √ | √ | Limited Reset for GTM |
| MSR_LM6 | √ | √ | √ | √ | Limited Reset for Ethernet |
| MSR_LM7 | √ | √ | √ | √ | Limited Reset for SENT |
| MSR_LM8 | √ | √ | √ | √ | Limited Reset for HS-USRT |
| MSR_LM10 | √ | √ | √ | √ | Limited Reset for CSI-H |
| MSR_LM11 | √ | √ | √ | √ | Limited Reset for RLIN3 |
| MSR_LM12 | √ | √ | √ | √ | Limited Reset for AD |

### 14.3.2 MSR_LM2 — Module Standby Register for ICUMC

This register is used to control the Stop Modes of the ICUMC.

This register is initialized by Power on reset, System reset 1, System Reset 2, Application reset 1 and Limited reset for ICUMC.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF8 6210$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MS_LM2 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.5 MSR_LM2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 0 | MS_LM2 | Setting this bit to 1 stops clock supply to ICUMC<br>0: ICUMC is operating.<br>1: Clock supply to ICUMC is stopped. |

It is forbidden to set MS_LM2 during SWLRESS2.SWLRESS2_0 = 1.

### 14.3.3 MSR_LM3 — Module Standby Register for MCAN

This register is used to control the Stop Modes of the MCAN.

This register is initialized by Power on reset, System reset 1, System Reset 2, Application reset 1 and Limited reset for MCAN.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF8 1710$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MS_LM3 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.6 MSR_LM3 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 0 | MS_LM3 | Setting this bit to 1 stops clock supply to MCAN<br>0: MCAN is operating.<br>1: Clock supply to MCAN is stopped. |

It is forbidden to set MS_LM3 during SWLRESS3.SWLRESS3_0 = 1

### 14.3.4 MSR_LM4 — Module Standby Register for FlexRay

This register is used to control the Stop Modes of the FlexRay.

This register is initialized by Power on reset, System reset 1, System Reset 2, Application reset 1 and Limited reset for FlexRay.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF8 $1810_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MS_LM4 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.7 MSR_LM4 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 0 | MS_LM4 | Setting this bit to 1 stops clock supply to FrexRay<br>0: FrexRay is operating.<br>1: Clock supply to FrexRay is stopped. |

It is forbidden to set MS_LM4 during SWLRESS4.SWLRESS4_0 = 1

### 14.3.5 MSR_LM5 — Module Standby Register for GTM

This register is used to control the Stop Modes of the GTM.

This register is initialized by Power on reset, System reset 1, System Reset 2, Application reset 1 and Limited reset for GTM.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF8 $1910_H$

**Value after reset:** 0000 $0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MS_LM5 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.8 MSR_LM5 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 0 | MS_LM5 | Setting this bit to 1 stops clock supply to GTM<br>0: GTM is operating.<br>1: Clock supply to GTM is stopped. |

It is forbidden to set MS_LM5 during SWLRESS5.SWLRESS5_0 = 1

### 14.3.6 MSR_LM6 — Module Standby Register for Ethernet

This register is used to control the Stop Modes of the Ethernet.

This register is initialized by Power on reset, System reset 1, System Reset 2, Application reset 1 and Limited reset for Ethernet.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF8 1A10$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MS_LM6 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.9 MSR_LM6 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 0 | MS_LM6 | Setting this bit to 1 stops clock supply to Ethernet<br>0: Ethernet is operating.<br>1: Clock supply to Ethernet is stopped. |

It is forbidden to set MS_LM6 during SWLRESS6.SWLRESS6_0 = 1

### 14.3.7 MSR_LM7 — Module Standby Register for SENT

This register is used to control the Stop Modes of the SENT.

This register is initialized by Power on reset, System reset 1, System Reset 2, Application reset 1 and Limited reset for SENT.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF8 $1B10_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MS_LM7 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.10 MSR_LM7 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 0 | MS_LM7 | Setting this bit to 1 stops clock supply to SENT<br>0: SENT is operating.<br>1: Clock supply to SENT is stopped. |

It is forbidden to set MS_LM7 during SWLRESS7.SWLRESS7_0 = 1

### 14.3.8 MSR_LM8 — Module Standby Register for HS-USRT

This register is used to control the Stop Modes of the HS-USRT.

This register is initialized by Power on reset, System reset 1, System Reset 2, Application reset 1 and Limited reset for US-USRT.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF8 $1C10_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MS_LM8 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.11 MSR_LM8 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 0 | MS_LM8 | Setting this bit to 1 stops clock supply to HS-USRT<br>0: HS-USRT is operating.<br>1: Clock supply to HS-USRT is stopped. |

It is forbidden to set MS_LM8 during SWLRESS8.SWLRESS8_0 = 1

### 14.3.9 MSR_LM10 — Module Standby Register for CSI-H

This register is used to control the Stop Modes of the CSI-H.

This register is initialized by Power on reset, System reset 1, System Reset 2, Application reset 1 and Limited reset for CSI-H.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF8 1E10$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MS_LM10 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.12 MSR_LM10 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 0 | MS_LM10 | Setting this bit to 1 stops clock supply to CSI-H<br>0: CSI-H is operating.<br>1: Clock supply to CSI-H is stopped. |

It is forbidden to set MS_LM10 during SWLRESS10.SWLRESS10_0 = 1

### 14.3.10 MSR_LM11 — Module Standby Register for RLIN3

This register is used to control the Stop Modes of the RLIN3.

This register is initialized by Power on reset, System reset 1, System Reset 2, Application reset 1 and Limited reset for RLIN3.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF8 1F10$_H$

**Value after reset:** 0000 0001$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MS_LM11 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.13 MSR_LM11 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 0 | MS_LM11 | Setting this bit to 1 stops clock supply to RLIN3<br>0: RLIN3 is operating.<br>1: Clock supply to RLIN3 is stopped. |

It is forbidden to set MS_LM11 during SWLRESS11.SWLRESS11_0 = 1

### 14.3.11 MSR_LM12 — Module Standby Register for AD

This register is used to control the Stop Modes of the AD.

This register is initialized by Power on reset, System reset 1, System Reset 2, Application reset 1 and Limited reset for AD.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF8 $2010_H$

**Value after reset:** 0000 $0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MS_LM12 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 14.14 MSR_LM12 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. The write value should always be 0. |
| 0 | MS_LM12 | Setting this bit to 1 stops clock supply to AD<br>0: AD is operating.<br>1: Clock supply to AD is stopped. |

It is forbidden to set MS_LM12 during SWLRESS12.SWLRESS12_0 = 1

# Section 15 External Memory Controller (MEMC)

This section describes the External Memory Controller (MEMC) module.

The first subsection describes all the characteristics specific to the RH850/P1x-C such as channels, register base addresses, and input/output signal names.
The second and subsequent subsections describe characteristics of External Memory Controller.

## 15.1 External Memory Controller (MEMC) Overview

### 15.1.1 Number of Channels

The RH850/P1x-C includes one physical channel.

**Table 15.1 MEMC Channels**

| | External Memory Controller |
|---|---|
| Number of Channels | 1 |
| Name | MEMC |

### 15.1.2 Register Addresses

External Memory Controller register addresses are represented by an offset from the base address <MEMC_base>.

**Table 15.2 Register Base Address <MEMC_base>**

| Module | <MEMC_base> Address |
|---|---|
| MEMC | $1003\ 0000_H$ |

### 15.1.3 Clock Supply

The following clock input is supplied for the External Memory Controller module.

**Table 15.3 Clock Supply**

| Unit Name | Specification | Description |
|---|---|---|
| MEMC | CLK_HSB (High speed system clock) | H-BUS clock |

**Note:** The input frequency must be 80MHz

### 15.1.4 External Input/Output Signals

The following table shows the External Memory Controller input/output signals.

**Table 15.4 MEMC External Input / Output Signals**

| Pin Name | I/O | Function |
|---|---|---|
| MEMC0A[8:0] | Output | External memory address output |
| MEMC0CS[3:0]Z | Output | External memory chip select output |
| MEMC0WRZ | Output | Write strobe output for external |
| MEMC0RDZ | Output | Read strobe output for external |
| MEMC0D[7:0] | Input/Output | Data input/output<br>Input pin MEMC0D*n*I (n = 0 to 7)<br>Output pin MEMC0D*n*O (n = 0 to 7) |

# 15.2 Overview

## 15.2.1 Functional Overview

The external memory controller (= MEMC) provides an interface to connect external memories, such as SRAM, ASICs, etc. to the microcontroller. This controller includes four chip select areas whose wait time can be set independently for each chip select area. This external memory controller has various programmable wait functions that can be set for each chip select area.

MEMC has following features.

- External address space.
  - Supports CS0 to CS3 address space.
  - Possible to access with max 512B linear address for each CS address space.
- Data bus width
  - Fixed 8bits
- Wait state
  - Programmable wait cycle for
  - Each CS address space (CS0 to CS3)
  - Each read access and write access
- Data hold wait
  - Possible to insert a wait for the state following the rising edge of the write strobe signal in order to secure the hold time for the data write strobe.
- Idle cycle
  - Possible to be set independently after a read cycle or after a write cycle for each chip select area.
- Supported clock frequency
  - Only 20MHz

## 15.2.2 Block Diagram

**Figure 15.1** shows MEMC block diagram.

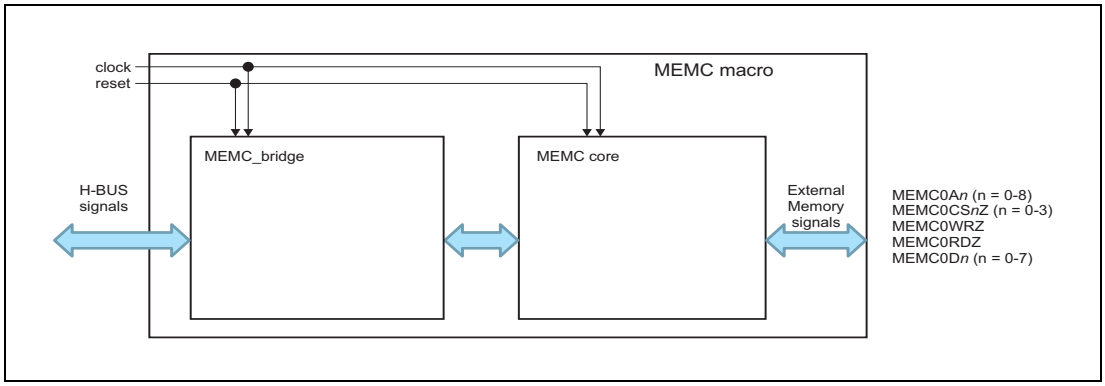



**Figure 15.1 MEMC block diagram**

## 15.3 Registers

These registers are used to control the setting of the external memory controller.

**Remark:** The clock counts shown this section indicate the clock count when operating on the external bus clock unless otherwise specified.

**Table 15.5 Overview of External Memory Controller Registers**

| Address Offset *1 | Register Name | Description | Access Width | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | Other |
| $04_H$ | MEMCBCT0 | Bus cycle type configuration register 0 | 16 | $0000_H$ | HB | — |
| $08_H$ | MEMCDWC0 | Data wait configuration register 0 | 16 | $FFFF_H$ | HB | — |
| $0C_H$ | MEMCDHC | Data hold wait configuration register | 16 | $0000_H$ | HB | — |
| $14_H$ | MEMCICC0 | Idle cycle configuration register 0 | 16 | $3333_H$ | HB | — |

Note 1. Base address is described in **Table 15.2**

**Table 15.6 Register Reset Condition**

| Register Name | Reset Condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset*1 |
| All registers | x | x | x | x | — |

Note 1. MEMC does not support Limited Reset.

### 15.3.1 MEMCBCT0 — Bus cycle type configuration register

The MEMCBCT0 enables/disables the generation of bus cycles for each chip select area. This register can be read or written in 16-bit units.

**CAUTION**

**Set the MEMCBCT0 register immediately after reset release, and do not change their values after they have been set. Otherwise, the operation will not be guaranteed.**

**Access:**

**Address:** Base + $04_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ME3 | — | BCT31 | BCT30 | ME2 | — | BCT21 | BCT20 | ME1 | — | BCT11 | BCT10 | ME0 | — | BCT01 | BCT00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W*1 | R | R/W | R/W | R/W*1 | R | R/W | R/W | R/W*1 | R | R/W | R/W | R/W*1 | R | R/W | R/W |

**Table 15.7 MEMCBCT0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 13, 12, 9, 8, 5, 4, 1, 0 | BCTn1, BCTn0 | Bus cycle setting bits<br>These bits set a memory (bus cycle type) to be connected to each chip select area in the separate bus mode.<br>**BCTn1 / BCTn0 / Memory type**<br>0 / 0 / SRAM<br>0 / 1 / Setting prohibited<br>1 / 0 / Setting prohibited<br>1 / 1 / Setting prohibited |
| 15, 11, 7, 3 | MEn | Memory controller operation enable bits<br>These bits enable or disable the operation of the internal memory controller for each chip select area.<br>**MEn / Memory controller operation**<br>0 / Operation stopped (External bus cycle is not generated.)<br>1 / Operation enabled<br>If the operation is disabled by the MEn bit, the generation of external bus cycles is stopped and the ongoing reading/writing is completed. |
| 14, 10, 6, 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

**Note:** n = 0 to 3

### 15.3.2 MEMCDWC0 — Data wait configuration register

The MEMCDWC0 register is used to set the number of data wait states of the external bus. These registers can be read or written in 16-bit units.

**Access:**

**Address:** Base + $08_H$

**Value after reset:** $FFFF_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DW33 | DW32 | DW31 | DW30 | DW23 | DW22 | DW21 | DW20 | DW13 | DW12 | DW11 | DW10 | DW03 | DW02 | DW01 | DW00 |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 15.8 MEMCDWC0 register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 15 to 12,<br>11 to 8,<br>7 to 4,<br>3 to 0 | DWm3, DWm2,<br>DWm1, DWm0 | Data wait setting bits<br>These bits set the number of data wait states for each chip select area. |

| DWm3 | DWm2 | DWm1 | DWm0 | Number of data wait states |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | No data wait |
| 0 | 0 | 0 | 1 | 1 clock |
| 0 | 0 | 1 | 0 | 2 clocks |
| 0 | 0 | 1 | 1 | 3 clocks |
| 0 | 1 | 0 | 0 | 4 clocks |
| 0 | 1 | 0 | 1 | 5 clocks |
| 0 | 1 | 1 | 0 | 6 clocks |
| 0 | 1 | 1 | 1 | 7 clocks |
| 1 | 0 | 0 | 0 | 8 clocks |
| 1 | 0 | 0 | 1 | 9 clocks |
| 1 | 0 | 1 | 0 | 10 clocks |
| 1 | 0 | 1 | 1 | 11 clocks |
| 1 | 1 | 0 | 0 | 12 clocks |
| 1 | 1 | 0 | 1 | 13 clocks |
| 1 | 1 | 1 | 0 | 14 clocks |
| 1 | 1 | 1 | 1 | 15 clocks |

**Note:** m = 0 to 3

### 15.3.3 MEMCDHC — Data hold wait configuration register

The MEMCDHC register is used to set the number of extended data hold wait states for each chip select area in the write cycle of the external bus. The number of data hold wait states determined by the set value of the MEMCDHC register + 1 cycle is inserted in a write cycle. This register can be read or written in 16-bit units.

**Access:** This register can be read or written in 16-bit units.

**Address:** Base + $0C_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DH31 | DH30 | DH21 | DH20 | DH11 | DH10 | DH01 | DH00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 15.9 MEMCDHC register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 15 to 8 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 7, 6, 5, 4, 3, 2, 1, 0 | DHn1, DHn0 | Data hold wait setting bits<br>These bits set the number of data hold wait states for each chip select area. |

| DHn1 | DHn0 | Number of data hold wait states |
|---|---|---|
| 0 | 0 | No data hold wait |
| 0 | 1 | 1 clock |
| 1 | 0 | 2 clocks |
| 1 | 1 | 3 clocks |

**Note:** n = 0 to 3

### 15.3.4 MEMCICC0 — Idle cycle configuration register

The MEMCICC0 register is used to set the number of idle cycles of the external bus. The number of idle cycles can be set for each chip select area and in the read cycle or write cycle. These registers can be read or written in 16-bit units.

**Access:**

**Address:** Base + $14_H$

**Value after reset:** $3333_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WIC31 | WIC30 | RIC31 | RIC30 | WIC21 | WIC20 | RIC21 | RIC20 | WIC11 | WIC10 | RIC11 | RIC10 | WIC01 | WIC00 | RIC01 | RIC00 |
| Value after reset | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 15.10 MEMCICC0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 13, 12, 9, 8, 5, 4, 1, 0 | RICn1, RICn0 | Idle cycle setting bits after read cycle<br>These bits set the number of idle cycles for each chip select area after a read cycle. |
| 15, 14, 11, 10, 7, 6, 3, 2, | WICn1, WICn0 | Idle cycle setting bits after write cycle<br>These bits set the number of idle cycles for each chip select area after a write cycle. |

| RICn1 | RICn0 | Number of idle cycles |
|---|---|---|
| 0 | 0 | No idle cycle*1 |
| 0 | 1 | 1 clock |
| 1 | 0 | 2 clocks |
| 1 | 1 | 3 clocks |

| WICn1 | WICn0 | Number of idle cycles |
|---|---|---|
| 0 | 0 | No idle cycle*1 |
| 0 | 1 | 1 clock |
| 1 | 0 | 2 clocks |
| 1 | 1 | 3 clocks |

Note 1. When the response cycle to CPU (or DMA) is inserted after it ends at the bus cycle, the interval that is few clocks spend on the response is caused regardless of the idle setting. Therefore, the interval of few clocks is caused when setting it to “No idle cycle” to the generation of the next bus cycle after it ends at the bus cycle. The additional idle cycles depends on an internal response & request latency.

**Note:** n = 0 to 3

## 15.4 Operation

### 15.4.1 Chip Select Output Function

The connected external memory area is managed divided into 4 chip select areas up to CSn (n = 0 to 3), as shown in **Figure 15.2**. The allocation of these chip select areas is fixed by the system and cannot be changed through programming. When a bus cycle is generated for the external bus, the external memory controller makes the MEMC0CS[3:0]Z output pins corresponding to the access target address active (low level), along with outputting the access target address from the MEMC0A[8:0] pins. The settings for the external bus, such as the number of wait/idle states, can be made for each chip select area.

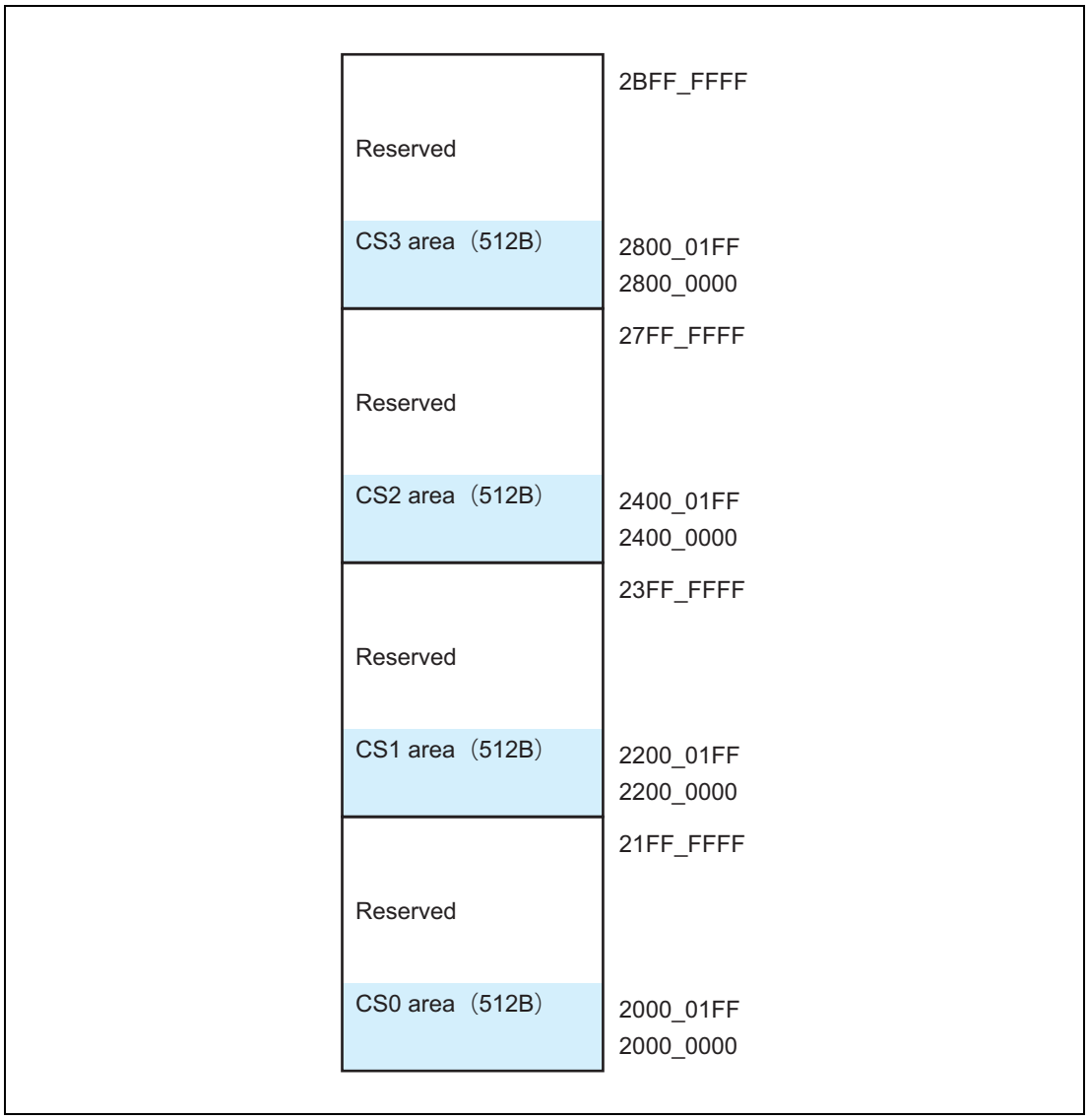



**Figure 15.2 External Memory Map**

### 15.4.2 Operation Enable/Operation Disable Setting Function

In this memory controller the operation can be enabled/disabled for each of the above-mentioned chip select areas shown in **Section 15.3.1** through the setting of the MEn (n = 0 to 3) bits of the MEMCBCT0 register.

If an access request is issued from the CPU (or DMA controller) to a chip select area for which operation has been disabled with this function, no external bus cycle is generated, the write value is ignored, and the read value becomes 0000 $0000_H$.

### 15.4.3 Wait functions

This external memory controller supports the wait functions listed below.

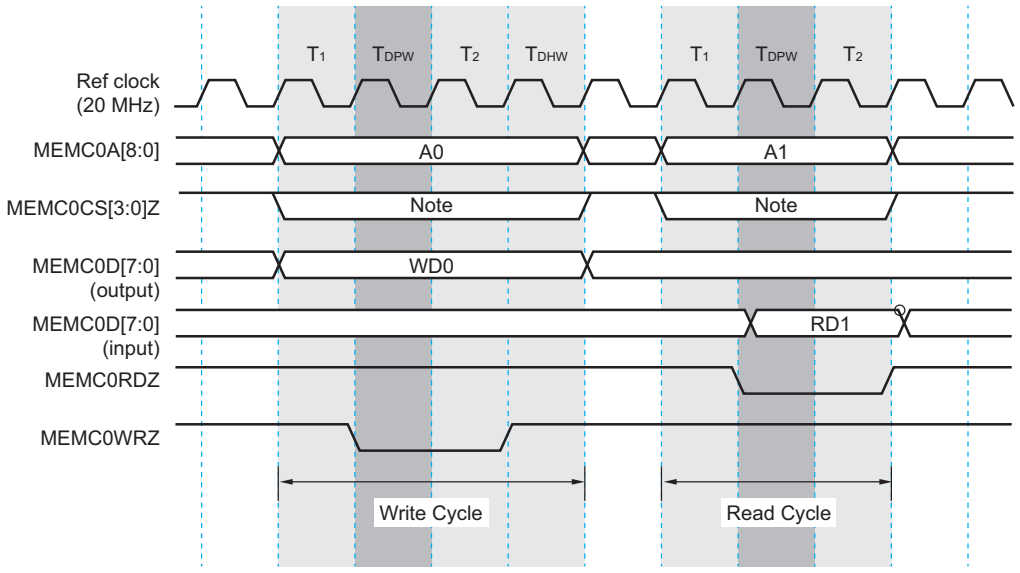
**Table 15.11 Wait Functions**

| Wait Function | | Programmable Data Wait | Data Hold Wait | Idle |
|---|---|---|---|---|
| SRAM bus cycle | Read | √ | — | √ |
| | Write | √ | √ | √ |
| Setting registers | | MEMCDWC0 | MEMCDHC0 | MEMCICC0 |
| Max. number of waits | | 15 | 3 | 3 |

### 15.4.4 Programmable Data Wait Function

This wait function is for delaying the data latch timing by extending the read strobe and write strobe periods. This function is enabled during the write accesses and at the first data transfer timing.

Up to 15 cycles can be inserted. Moreover, setting individual chip select areas with the MEMCDWC register is also possible.




**Note:** Indicates the chip select area to be accessed.

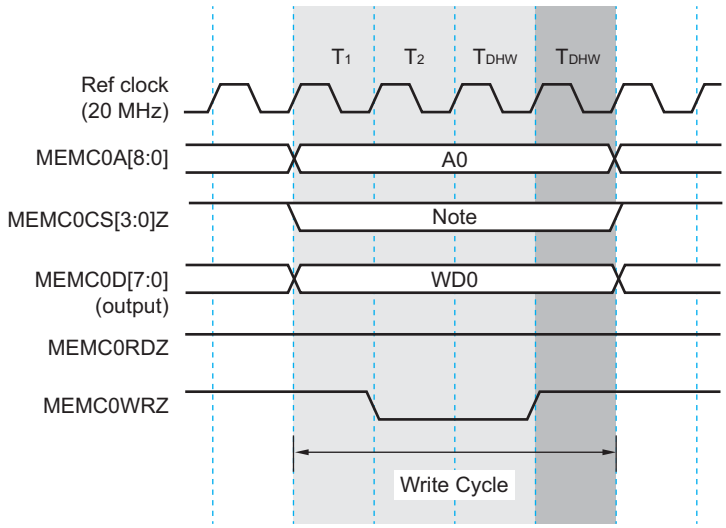
**Figure 15.3 Programmable Data Wait**

### 15.4.5 Data Hold Wait Function

This function inserts a wait for the state following the rising edge of the write strobe signal in order to secure the hold time for the data write strobe. This function is enabled only during the write cycle

This external memory controller always inserts 1 data hold wait state upon occurrence of a write cycle. This data hold wait extends the MEMCDHC register setting by up to 3 cycles, allowing insertion of 4 cycles.

The number of data hold wait extensions can be set for each chip select area with the MEMCDHC register. The initial status is no wait extension for any of the chip select areas (1 data hold wait cycle).




**Note:** Indicates the chip select area to be accessed.

**Figure 15.4 Data Hold Wait**

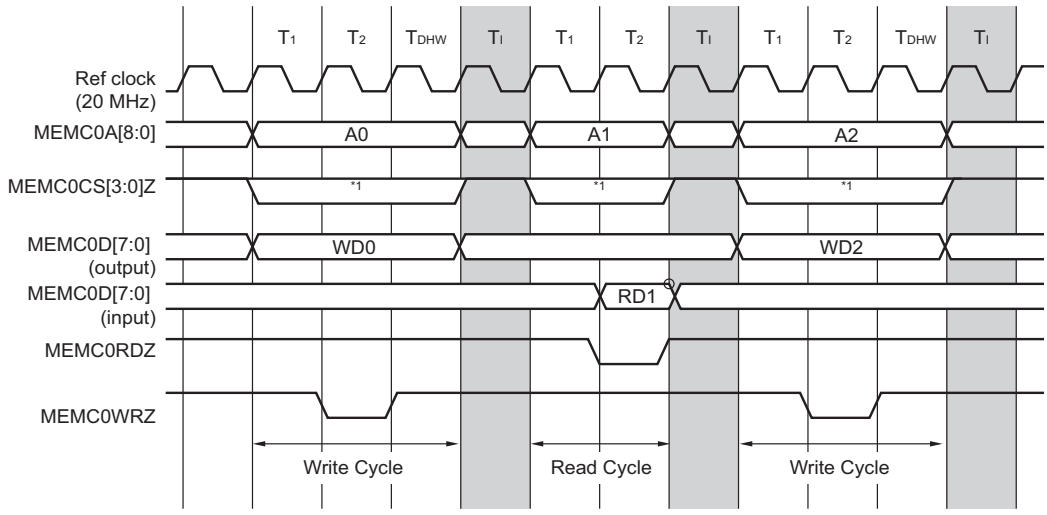
### 15.4.6 Idle Insertion Function

This function inserts an idle state after the last state of each cycle in order to prevent bus fights between cycles. For example, if a 32-bit write access occurs when an external memory that has an 8-bit bus width is connected, four write cycles are executed, and then an idle state is inserted.

This function can be set independently after a read cycle or after a write cycle for each chip select area by setting the ICC register. Up to 3 cycles can be inserted.

**CAUTION**

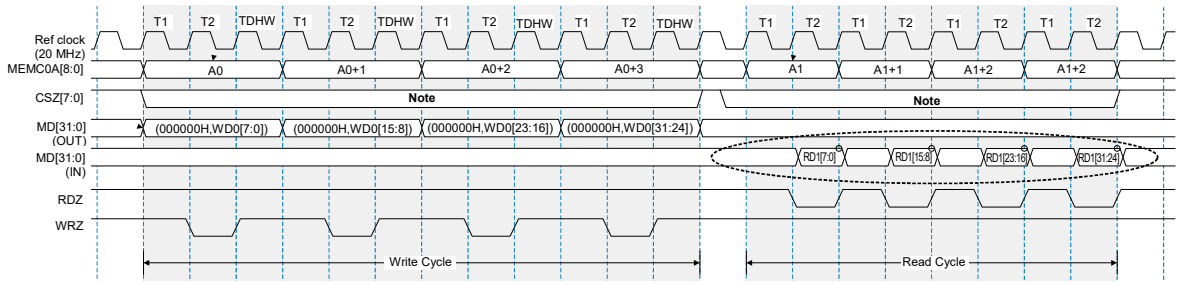
**When the response cycle to CPU (or DMA) is inserted after it ends at the bus cycle, the interval that is few clocks spend on the response is caused regardless of the idle setting. Therefore, the interval of few clocks is caused when setting it to "No idle cycle" to the generation of the next bus cycle after it ends at the bus cycle. The additional idle cycles depends on an internal response & request latency.**




**Note:** Indicates the chip select area to be accessed

**Figure 15.5 Idle Cycle**

### 15.4.7 Memory access timing examples




**Note:** Indicates the chip select area to be accessed.

Note 1. If the data is wider than 32 bits, separate 32 bit unit and transfer little endian manner

**Figure 15.6 Memory access timing example, 32-bit Read/Write, 8-bit External Bus (see *1)**

## 15.5 Difference among P1M-C, P1H-C and P1H-CE

Table 15.12 Difference of number of macro

| Macro | Device | | | | | |
|---|---|---|---|---|---|---|
| | P1M-C (QFP, BGA-292) | P1M-C (BGA-156) | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
| MEMC | — | — | 1 | — | 1 | 1 |

# Section 16 Clocked Serial Interface H (CSIH)

This section contains a generic description of the Clocked Serial Interface H (CSIH).

The first part of this section describes all RH850/P1x-C specific properties, such as the number of units, register base addresses, etc.

The remainder of the section describes the functions and registers of CSIH.

## 16.1 Features of RH850/P1x-C CSIH

This module is the clock serial interface (CSIH) with the following additional functions.

- Flexible idle state control mode
- Silent mode
- Sample point modification function
- Priority management module (PMM) function

As the priority management module function is a relatively large additional function within the above functions, it will be described individually following the description on the basic functions of CSIH.

### 16.1.1 Units

This microcontroller has the following number of CSIH units.

Each CSIH unit has one channel interface.

**Table 16.1 Units**

| | CSIH |
|---|---|
| Product Name | P1M-C, P1H-C and P1H-CE |
| Number of Units | 4 |
| Name | CSIHn (n = 0 to 3) |

**Table 16.2 Unit Configurations and Channels**

| Unit Name (Channel Name) CSIHn | Channels per Unit | RH850/P1x-C (4 ch) |
|---|---|---|
| CSIH0 | 1 | √ |
| CSIH1 | 1 | √ |
| CSIH2 | 1 | √ |
| CSIH3 | 1 | √ |

**Note:** The channel names are same as those of the corresponding units.

**Table 16.3 Index (1/2)**

| Index | Meaning |
|---|---|
| n | Throughout this section, the individual CSIH units are identified by the index “n” (n = 0 to 3): for example, CSIHnCTL0 is the CSIHn control register0. |
| x | CSIHn has up to 8 chip select signals. Throughout this section, the individual chip select signals are identified by the index “x” (x = 0 to 7): that is, CSx denotes a non-specified chip select signal. |

Table 16.3 Index (2/2)

| Index | Meaning |
|---|---|
| y | A variable used for explanation is identified by the index “y” (y = 0 to 3): for example, CSIHnBRSy is a non-specified baud rate setting register of CSIHn. |
| z | The clock division ratio used in calculating the bit rate is indicated by the letter “z”. |
| k | The value of the bit rate setting is indicated by the letter “k”. |

Table 16.4 Number of Chip Select Signals

| Unit Name | Chip Select Index |
|---|---|
| CSIH0 | CSIH0CSSx (x = 0 to 7) |
| CSIH1 | CSIH1CSSx (x = 0 to 7) |
| CSIH2 | CSIH2CSSx (x = 0 to 7) |
| CSIH3 | CSIH3CSSx (x = 0 to 7) |

## 16.1.2 Register Base Address

CSIH base addresses are listed in the following table.

CSIH register addresses are given as offsets from the base addresses in general.

Table 16.5 Register Base Address

| Base Address Name | Base Address |
|---|---|
| <CSIH0_base> | FFD8 $0000_H$ |
| <CSIH1_base> | FFCA $0000_H$ |
| <CSIH2_base> | FFD8 $3000_H$ |
| <CSIH3_base> | FFCA $3000_H$ |

## 16.1.3 Clock Supply

Clock supply by and to CSIH is listed in the following table.

Table 16.6 Clock Supply

| Unit Name | Clock for the Unit | Supply Clock Name |
|---|---|---|
| CSIHn | P-Bus interface clock (PCLK) | CLK_HSB |
| | CSIHn communication clock (clkc) | CLKP_C |

**Note:** In case of P1M-C, P1H-C following clock condition must be fulfilled
CLK_HSB < CLKP_C < (3 x CLK_HSB)
In case of P1H-CE following clock condition must be fulfilled
CLK_HSB <= CLKP_C < (3 x CLK_HSB)

For detail of clock supply, see **Section 12, Clock Controller**.

### 16.1.4 Interrupt Requests

CSIH interrupt requests are listed in the following table.

**Table 16.7 Interrupt Requests**

| Unit Interrupt Name | Outline | Interrupt Number | DMA Trigger Number |
|---|---|---|---|
| **CSIH0** | | | |
| CSIHTIC | Communication status interrupt | 88 | 79 |
| CSIHTIR | Receive status interrupt | 87 | 78 |
| CSIHTIRE | Communication error interrupt | 86 | — |
| CSIHTIJC | Job completion interrupt | 89 | 80 |
| **CSIH1** | | | |
| CSIHTIC | Communication status interrupt | 92 | 82 |
| CSIHTIR | Receive status interrupt | 91 | 81 |
| CSIHTIRE | Communication error interrupt | 90 | — |
| CSIHTIJC | Job completion interrupt | 93 | 83 |
| **CSIH2** | | | |
| CSIHTIC | Communication status interrupt | 96 | 85 |
| CSIHTIR | Receive status interrupt | 95 | 84 |
| CSIHTIRE | Communication error interrupt | 94 | — |
| CSIHTIJC | Job completion interrupt | 97 | 86 |
| **CSIH3** | | | |
| CSIHTIC | Communication status interrupt | 100 | 88 |
| CSIHTIR | Receive status interrupt | 99 | 87 |
| CSIHTIRE | Communication error interrupt | 98 | — |
| CSIHTIJC | Job completion interrupt | 101 | 89 |

### 16.1.5 Hardware Reset

CSIH reset sources are listed in the following table. CSIH is initialized by these reset sources.

**Table 16.8 Reset Sources**

| | Reset condition | | | | |
|---|---|---|---|---|---|
| Reset Name | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| All Registers | √ | √ | √ | √ | √ |

### 16.1.6 External Input/Output Signals

External input/output signals of CSIH are listed below.

**Table 16.9 External Input/Output Signals**

| Unit Signal Name | Outline | Alternative port pin signal |
|---|---|---|
| **CSIH0** | | |
| CSIHTSCK(in) | Serial clock input signal | CSIH0SCI |
| CSIHTSCK(out) | Serial clock output signal | CSIH0SCO |
| CSIHTSI | Serial data input signal | CSIH0SI |
| CSIHTSSIZ | SS function control input signal | CSIH0SSIZ |
| CSIHTRYI | Ready(1)/busy(0) input signal | CSIH0RYI |
| CSIHTSO | Serial data output signal | CSIH0SO |
| CSIHTRYO | Ready(1)/busy(0) output signal | CSIH0RYO |
| CSIHTCSS[7:0] | Serial peripheral chip select signal | CSIH0CSS[7:0] |
| CSIHTDCS | Data consistency check signal | CSIH0DCS |
| **CSIH1** | | |
| CSIHTSCK(in) | Serial clock input signal | CSIH1SCI |
| CSIHTSCK(out) | Serial clock output signal | CSIH1SCO |
| CSIHTSI | Serial data input signal | CSIH1SI |
| CSIHTSSIZ | SS function control input signal | CSIH1SSIZ |
| CSIHTRYI | Ready(1)/busy(0) input signal | CSIH1RYI |
| CSIHTSO | Serial data output signal | CSIH1SO |
| CSIHTRYO | Ready(1)/busy(0) output signal | CSIH1RYO |
| CSIHTCSS[7:0] | Serial peripheral chip select signal | CSIH1CSS[7:0] |
| CSIHTDCS | Data consistency check signal | CSIH1DCS |
| **CSIH2** | | |
| CSIHTSCK(in) | Serial clock input signal | CSIH2SCI |
| CSIHTSCK(out) | Serial clock output signal | CSIH2SCO |
| CSIHTSI | Serial data input signal | CSIH2SI |
| CSIHTSSIZ | SS function control input signal | CSIH2SSIZ |
| CSIHTRYI | Ready(1)/busy(0) input signal | CSIH2RYI |
| CSIHTSO | Serial data output signal | CSIH2SO |
| CSIHTRYO | Ready(1)/busy(0) output signal | CSIH2RYO |
| CSIHTCSS[7:0] | Serial peripheral chip select signal | CSIH2CSS[7:0] |
| CSIHTDCS | Data consistency check signal | CSIH3DCS |
| **CSIH3** | | |
| CSIHTSCK(in) | Serial clock input signal | CSIH3SCI |
| CSIHTSCK(out) | Serial clock output signal | CSIH3SCO |
| CSIHTSI | Serial data input signal | CSIH3SI |
| CSIHTSSIZ | SS function control input signal | CSIH3SSIZ |
| CSIHTRYI | Ready(1)/busy(0) input signal | CSIH3RYI |
| CSIHTSO | Serial data output signal | CSIH3SO |
| CSIHTRYO | Ready(1)/busy(0) output signal | CSIH3RYO |
| CSIHTCSS[7:0] | Serial peripheral chip select signal | CSIH3CSS[7:0] |
| CSIHTDCS | Data consistency check signal | CSIH3DCS |

### 16.1.7 Data Consistency Check

The following table lists the port pins on which CSIHnSO pin functions are multiplexed and whether or not the CSIHnSO pin functions support data consistency checking. See **Section 16.5.12, Error detection** for details on data consistency checking.

**Table 16.10 Data Consistency Checking and Port Pins**

| Unit Signal Name | Port Pin Name | Alternative Function | Data Consistency Checking |
|---|---|---|---|
| **CSIH0** | | | |
| CSIHTSO | P2_13 | ALT_OUT3 | Supported (except P1M-C (BGA-156)/P1H-C (4MB, BGA-156)) |
| | P0_9 | ALT_OUT3 | Supported (except P1M-C (BGA-156)/P1H-C (4MB, BGA-156)) |
| | P5_5 | ALT_OUT2 | Supported |
| **CSIH1** | | | |
| CSIHTSO | P4_2 | ALT_OUT3 | Supported |
| | P0_7 | ALT_OUT4 | Supported (except P1M-C (BGA-156)/P1H-C (4MB, BGA-156)) |
| | P1_2 | ALT_OUT3 | Supported |
| **CSIH2** | | | |
| CSIHTSO | P2_1 | ALT_OUT3 | Supported |
| | P4_10 | ALT_OUT3 | Supported |
| | P5_6 | ALT_OUT4 | Supported |
| **CSIH3** | | | |
| CSIHTSO | P8_0 | ALT_OUT4 | Supported (except P1M-C/P1H-C (4MB, BGA-156)) |
| | P1_4 | ALT_OUT2 | Supported |
| | P6_13 | ALT_OUT2 | Supported (except P1M-C (QFP-144)/P1M-C (BGA-156)/ P1H-C (4MB, BGA-156)) |

**CAUTION**

**PUCCn_m bit in PUCCn register and PDSCn_m bit in PDSCn register used for CSIH data pins should be set as written below depending on the communication speed and load capacitance (C load). For detail, See the Pin function.**

| Communication speed | C load | Register setting |
|---|---|---|
| Faster than 10 Mbps | Up to 15pF | PUCCn_m = 1, and PDSCn_m = 1 |
| Slower than 10Mbps | Up to 100pF | PUCCn_m = 1, and PDSCn_m = 1 |
| | up to 30pF | PUCCn_m = 1, and PDSCn_m = 0/1 |
| | Up to 20pF | PUCCn_m = 1 and PDSCn_m = 0/1 |
| | | PUCCn_m = 0 and PDSCn_m = 1 |

## 16.2 Overview

### 16.2.1 Functional Overview

- Three-wire serial synchronous data transfer
- Full duplex operation (simultaneous transfer and receive), receive only mode, or transmit only mode
- Master mode and slave mode selectable
- Phase of clock and data selectable for each chip select
- Data transfer with MSB or LSB first selectable for each chip select
- Transfer data length selectable from 2 to 16 bits in 1-bit units for each chip select
- EDL (Extended Data Length) function for transferring data with more than 16 bits
- Maximum transmission speed:
  - in master mode: 20 MHz
  - in slave mode: 20 MHz
- Bit rate selectable by BRG (baud rate generator) output (at Master mode) or by slave clock
- Transmit mode, Receive mode and Transmit/Receive mode selectable
- Buffer size is 128 words (1 word is data 32 bits + ECC 7 bits)
- Memory mode selectable (FIFO, dual buffer, transmit-only buffer, and direct access)
- Built-in handshake function
- Error detection (data consistency check, parity, time-out, overflow, and overrun)
- JOB enable control bit for AUTOSAR
- RCB ( Recessive Configuration for Broadcasting ) bit for Broadcasting
- LBM (Loop Back Mode) function for self-test
- Four different interrupt request signals. (INTCSIHnTIC, INTCSIHnTIR, INTCSIHnTIRE, INTCSIHnTIJC)
- IDLE State Control function.
- Silent mode communication for extended idle time
- Automatically generation of chip select output signal with configurable active level
- Data transfer without activated chip select
- Transmission speed for each chip select is selectable out of four predefined baud rates (in master mode) or by clock input signal from master (in slave mode)
- Full DMA support for all CSIH registers (The SPI interface should be accessible by more than one bus-master without explicit SW-synchronization)

### 16.2.2 Functional Overview Description

The Clocked Serial Interface uses three signals for communication:

- Transmission clock CSIHTSCK (output in master mode, input in slave mode)
- Data output signal CSIHTSO
- Data input signal CSIHTSI

Additional signals are available for external control and monitoring.

- CSIHTSSIZ: Slave select input signal
- CSIHTRYO: Ready/busy output signal (handshake signal)
- CSIHTRYI: Ready/busy input signal (handshake signal)
- CSIHTCSS[7:0]: Chip select signals
- CSIHTDCS: Data consistency check signal

Data transmission is bit-wise and serial, and synchronous to the transmission clock.

The following table shows the most important registers for setting up the CSIH.

**Table 16.11 Most important registers for setting up the CSIH**

| Register | Function |
|---|---|
| CSIHnCTL0 | Enables/disables transmission clock, and permits/ prohibits data transmission and data reception. Defines end-of-job behavior and enables/disables buffering (bypass of the buffer). |
| CSIHnCTL1 | Controls options like interrupt timing, extended data length, job feature, data consistency check, loop-back mode, handshake, etc. |
| CSIHnCTL2 | Selects master/slave mode, and the baud rate of the internal generator (BRG) in master mode |
| CSIHnMCTL0 | Selects memory mode and specifies the time-out value |
| CSIHnMCTL1 | Controls the memory in FIFO mode |
| CSIHnMCTL2 | Controls the memory in dual buffer mode |
| CSIHnCFGx | Registers to configure the communication protocol for each chip select signal |

### 16.2.3 Block Diagram

The block diagram shows the main components of the CSIH.

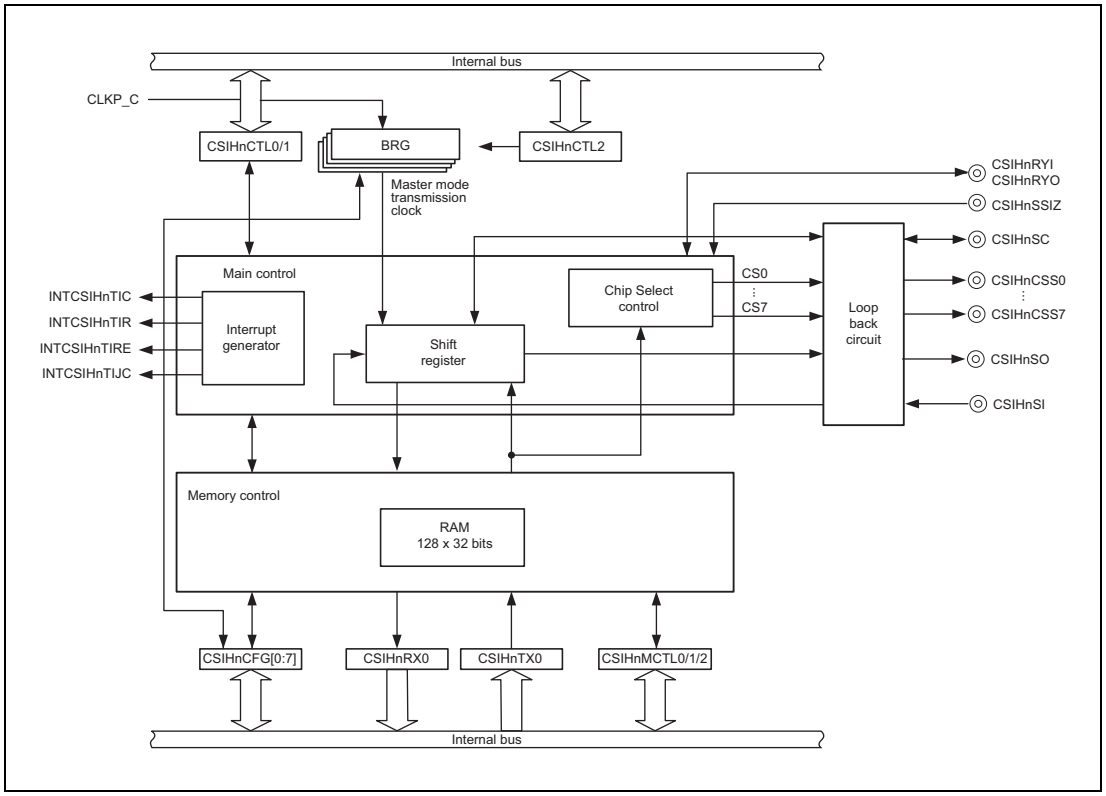



**Figure 16.1 CSIH block diagram**

In master mode, the transmission clock CSIHTSCK is generated by the built-in baud rate generator (BRG). In slave mode, the transmission clock is provided from an external source.

The built-in memory can be configured as FIFO, dual buffer (separate transmit and receive buffers), or transmit-only buffer. It can also be bypassed for data transmission and reception without buffering.

The loop back circuit disconnects the CSIH completely from the ports and supports internal self-test.

It is possible to pre-configure 4 different baud rates, which can be individually selected for each CS.

**NOTE**

This chapter describes the following modes:

- The “operating mode” separates between master and slave mode. In this context, only a master can control and communicate with several slaves (for details see **Section 16.5.1, Operating modes (master/slave)**).
- The “job mode” is related to the AUTOSAR job concept (for details see **Section 16.5.3.3, Job concept**).
- The “memory mode” takes the various configurations of the associated buffer memory into account (for details see **Section 16.5.6, CSIH buffer memory**).
- The “data transfer mode” specifies the kind of the communication — transmit-only, receive only, or transmit/receive (for details see **Section 16.5.7, Data transfer modes**).

# 16.3 Registers

## 16.3.1 List of Registers

CSIH registers are listed in the following table.

**Table 16.12 Registers (1/2)**

| Address <CSIHn_base> + | Register Name | Description | Access Width | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | other |
| $0000_H$ | CSIHnCTL0 | CSIH control register 0 | 8 | $00_H$ | *1 | — |
| $0010_H$ | CSIHnCTL1 | CSIH control register 1 | 32 | $0000\ 0000_H$ | *1 | — |
| $0014_H$ | CSIHnCTL2 | CSIH control register 2 | 16 | $E000_H$ | *1 | — |
| $1000_H$ | CSIHnMCTL1 | CSIH memory control register 1 | 32 | $0000\ 0000_H$ | *1 | — |
| $1004_H$ | CSIHnMCTL2 | CSIH memory control register 2 | 32 | $0000\ 0000_H$ | *1 | — |
| $1008_H$ | CSIHnTX0W | CSIH transmit data register 0 | 32 | — | *1 | — |
| $100C_H$ | CSIHnTX0H | CSIH transmit data register 0 | 16 | — | *1 | — |
| $1010_H$ | CSIHnRX0W | CSIH receive data register 0 | 32 | — | *1 | — |
| $1014_H$ | CSIHnRX0H | CSIH receive data register 0 | 16 | — | *1 | — |
| $1018_H$ | CSIHnMRWP0 | CSIH memory read/write pointer register 0 | 32 | $0000\ 0000_H$ | *1 | — |
| $101C_H$ | CSIHnSTR0 | CSIH status register 0 | 32 | $0000\ 0010_H$ | *1 | — |
| $1020_H$ | CSIHnSTCR0 | CSIH status clear register 0 | 16 | $0000_H$ | *1 | — |
| $1040_H$ | CSIHnMCTL0 | CSIH memory control register 0 | 16 | $001F_H$ | *1 | — |
| $1044_H$ | CSIHnCFG0 | CSIH configuration register 0 | 32 | $0000\ 0000_H$ | *1 | — |
| $1048_H$ | CSIHnCFG1 | CSIH configuration register 1 | 32 | $0000\ 0000_H$ | *1 | — |
| $104C_H$ | CSIHnCFG2 | CSIH configuration register 2 | 32 | $0000\ 0000_H$ | *1 | — |
| $1050_H$ | CSIHnCFG3 | CSIH configuration register 3 | 32 | $0000\ 0000_H$ | *1 | — |
| $1054_H$ | CSIHnCFG4 | CSIH configuration register 4 | 32 | $0000\ 0000_H$ | *1 | — |
| $1058_H$ | CSIHnCFG5 | CSIH configuration register 5 | 32 | $0000\ 0000_H$ | *1 | — |
| $105C_H$ | CSIHnCFG6 | CSIH configuration register 6 | 32 | $0000\ 0000_H$ | *1 | — |
| $1060_H$ | CSIHnCFG7 | CSIH configuration register 7 | 32 | $0000\ 0000_H$ | *1 | — |
| $1068_H$ | CSIHnBRS0 | CSIH baud rate setting register 0 | 16 | $0000_H$ | *1 | — |
| $106C_H$ | CSIHnBRS1 | CSIH baud rate setting register 1 | 16 | $0000_H$ | *1 | — |
| $1070_H$ | CSIHnBRS2 | CSIH baud rate setting register 2 | 16 | $0000_H$ | *1 | — |
| $1074_H$ | CSIHnBRS3 | CSIH baud rate setting register 3 | 16 | $0000_H$ | *1 | — |
| $2000_H$ | PMMAnCTL | PMM control register | 32 | $0000\ 0000_H$ | *1 | — |
| $2004_H$ | PMMAnTCTL0 | PMM TG0 control register | 32 | $0000\ 0000_H$ | *1 | — |
| $2008_H$ | PMMAnTCTL1 | PMM TG1 control register | 32 | $0000\ 0000_H$ | *1 | — |
| $200C_H$ | PMMAnTCTL2 | PMM TG2 control register | 32 | $0000\ 0000_H$ | *1 | — |
| $2010_H$ | PMMAnTCTL3 | PMM TG3 control register | 32 | $0000\ 0000_H$ | *1 | — |
| $2014_H$ | PMMAnTCTL4 | PMM TG4 control register | 32 | $0000\ 0000_H$ | *1 | — |
| $2018_H$ | PMMAnTCTL5 | PMM TG5 control register | 32 | $0000\ 0000_H$ | *1 | — |

Table 16.12 Registers (2/2)

| Address <CSIHn_base> + | Register Name | Description | Access Width | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | other |
| $201C_H$ | PMMAnTCTL6 | PMM TG6 control register | 32 | $0000\ 0000_H$ | *1 | — |
| $2020_H$ | PMMAnTCTL7 | PMM TG7 control register | 32 | $0000\ 0000_H$ | *1 | — |
| $2024_H$ | PMMAnTTX0 | PMM TG0 transmit buffer | 32 | $0000\ 0000_H$ | *1 | — |
| $2028_H$ | PMMAnTTX1 | PMM TG1 transmit buffer | 32 | $0000\ 0000_H$ | *1 | — |
| $202C_H$ | PMMAnTTX2 | PMM TG2 transmit buffer | 32 | $0000\ 0000_H$ | *1 | — |
| $2030_H$ | PMMAnTTX3 | PMM TG3 transmit buffer | 32 | $0000\ 0000_H$ | *1 | — |
| $2034_H$ | PMMAnTTX4 | PMM TG4 transmit buffer | 32 | $0000\ 0000_H$ | *1 | — |
| $2038_H$ | PMMAnTTX5 | PMM TG5 transmit buffer | 32 | $0000\ 0000_H$ | *1 | — |
| $203C_H$ | PMMAnTTX6 | PMM TG6 transmit buffer | 32 | $0000\ 0000_H$ | *1 | — |
| $2040_H$ | PMMAnTTX7 | PMM TG7 transmit buffer | 32 | $0000\ 0000_H$ | *1 | — |
| $2044_H$ | PMMAnRX | PMM TG receive buffer | 32 | $0000\ 0000_H$ | *1 | — |
| $2048_H$ | PMMAnTCNT0 | PMM TG0 counter | 32 | $0000\ 0000_H$ | *1 | — |
| $204C_H$ | PMMAnTCNT1 | PMM TG1 counter | 32 | $0000\ 0000_H$ | *1 | — |
| $2050_H$ | PMMAnTCNT2 | PMM TG2 counter | 32 | $0000\ 0000_H$ | *1 | — |
| $2054_H$ | PMMAnTCNT3 | PMM TG3 counter | 32 | $0000\ 0000_H$ | *1 | — |
| $2058_H$ | PMMAnTCNT4 | PMM TG4 counter | 32 | $0000\ 0000_H$ | *1 | — |
| $205C_H$ | PMMAnTCNT5 | PMM TG5 counter | 32 | $0000\ 0000_H$ | *1 | — |
| $2060_H$ | PMMAnTCNT6 | PMM TG6 counter | 32 | $0000\ 0000_H$ | *1 | — |
| $2064_H$ | PMMAnTCNT7 | PMM TG7 counter | 32 | $0000\ 0000_H$ | *1 | — |
| $2068_H$ | PMMAnSTR0 | PMM TG status register 0 | 32 | $0000\ 0000_H$ | *1 | — |
| $206C_H$ | PMMAnSTR1 | PMM TG status register 1 | 32 | $0000\ 0000_H$ | *1 | — |
| $2070_H$ | PMMAnSTC | PMM TG status clear register | 32 | $0000\ 0000_H$ | *1 | — |

Note 1. In the case of
n = 0 PBG3#1.PG3-CSIH0,
n = 1 PBG1#1.PG1-CSIH1,
n = 2 PBG3#1.PG3-CSIH2,
n = 3 PBG1#1.PG1-CSIH3

### <CSIHn_base>

The base address <CSIHn_base> of the CSIHn is defined in **Section 16.1.2, Register Base Address**.

### 16.3.2 CSIHnCTL0 — CSIH control register 0

This register controls the operation clock and enables/disables transmission/reception and the memory part for transmission and/or reception. It forces the stop of communication at the end of the current job.

**Access:** This register can be read/write in 8-bit or 1-bit units.

**Address:** <CSIHn_base> + $0000_H$

**Value after reset:** $00_H$ This register is initialized by reset operation.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CSIHnPWR | CSIHnTXE | CSIHnRXE | — | — | — | CSIHnJOBE | CSIHnMBS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R | R | R | R/W | R/W |

**Table 16.13 CSIHnCTL0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | CSIHnPWR | Controls the operation clock.<br>0: Stops operation clock0.<br>1: Provides operation clock.<br>Clearing CSIHnPWR to 0 resets the internal circuits, stops operation, and sets CSIH to standby state. No clock is provided to internal circuits.<br><br>If CSIHnPWR is cleared during communication, ongoing communication is immediately aborted. In this case, communication must be started over. |
| 6 | CSIHnTXE | Enables/disables transmission.<br>0: Transmission disabled<br>1: Transmission enabled |
| 5 | CSIHnRXE | Enables/disables reception.<br>0: Reception disabled<br>1: Reception enabled. |
| 4 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | CSIHnJOBE | Stops communication at the end of the current job (Communication ends after data with CSIHnEOJ = 1 was sent.):<br>0: Communication stop is not requested.<br>1: Stops communication.<br>This bit can be used to abort an ongoing job. This bit is cleared to 0 automatically. If this bit is set to 1, the read value is always 0.<br>In FIFO mode, the pointer must be cleared by setting CSIHnSTCR0.CSIHnPCT = 1 before the next communication is started<br>This bit is disabled when CSIHnCTL1.CSIHnJE = 0. |
| 0 | CSIHnMBS | Bypasses the memory for transmission and/or reception data.<br>0: Memory mode<br>CSIH memory is used for transmission and/or reception data.<br>1: Direct access mode<br>CSIH memory is bypassed.<br>**CAUTION**<br>High priority communication function based on CPU's control is executing, behavior of the macro is as same as direct access mode regardless of CSIHnMBS. |

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

### 16.3.3 CSIHnCTL1 — CSIH control register 1

This register specifies the interrupt timing and the interrupt delay mode. It enables/disables extended data length control, data consistency check, loop-back mode, handshake functionality, and job mode. It selects the active output level of each chip select signal and the behavior of the chip select signals after the transfer of the final data.

**Access:** This register can be read/write in 32-bit units.

**Address:** <CSIHn_base> + $0010_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CSIHn SLRS | — | — | — | CSIHn ISCE | CSIHn SME | — | CSIHn CKR | CSIHn SLIT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R | R | R | R/W | R/W | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHn CSL7 | CSIHn CSL6 | CSIHn CSL5 | CSIHn CSL4 | CSIHn CSL3 | CSIHn CSL2 | CSIHn CSL1 | CSIHn CSL0 | CSIHn EDLE | CSIHn JE | CSIHn DCS | CSIHn CSRI | CSIHn LBM | CSIHn SIT | CSIHn HSE | CSIHn SSE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.14 CSIHnCTL1 register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 25 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 24 | CSIHnSLRS | Sets the internal sampling time for receive data input.<br>0: Standard sampling point<br>1: Shifted sampling point<br>In master mode, this bit relaxes the setup time of received data by half clock period of communication clock. In slave mode, this bit is invalid.<br>When The communication speed is faster than 5Mbps (excluding 5Mbps), CSIHnSLRS bit should be set to 1.<br>For detail about sampling point, see **Table 16.29, CSIHnCFGx register contents (3/5 and 4/5)**. |
| 23 to 21 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 20 | CSIHnISCE | Idle state control mode enable/disable bit<br>0: ISC mode disabled<br>The behavior between two transfers will be controlled by the setting in CSIHnCFGx.CSIHnIDLx bit.<br>1: ISC mode enabled<br>The behavior between two transfers will be controlled by the setting in CSIHnTX0W.CSIHnISC bit.<br>The CSIHnCFGx.CSIHnIDLx bit must be set to 0 if idle state control mode is enabled.<br>This bit can only be written if CSIHnCTL0.CSIHnPWR = 0 |
| 19 | CSIHnSME | Enable/disable silent mode.<br>0: Disables silent mode<br>1: enables silent mode |
| 18 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 17 | CSIHnCKR | CSIHSCK clock inversion function<br>0: The default level of CSIHSC is high<br>1: The default level of CSIHSC is low<br>For details, see **16.3.15, CSIHnCFGx — CSIH Configuration register (x = 0 to 7)**. |

**Table 16.14 CSIHnCTL1 register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 16 | CSIHnSLIT | Selects the timing of interrupt INTCSIHnTIC.<br>0: Normal interrupt timing (interrupt is generated after the transfer)<br>1: As soon as the contents of the CSIHnTX0 register are transferred to the shift register, an interrupt is generated (this function is activated only in direct access mode).<br>For details, see **Section 16.4.3, CSIHTIC (communication interrupt)**.<br>**CAUTION**<br>SLIT setting is invalid in the TX Only Buffer mode. |
| 15 to 8 | CSIHnCSLx (x = 0 to 7) | Selects the active output level of chip select signal x (CSIHnCSSx).<br>0: Chip select is active low.<br>1: Chip select is active high.<br>For details, see **Section 16.5.3, Chip selection (CS) features**. |
| 7 | CSIHnEDLE | Enables/disables extended data length (EDL) mode.<br>0: Disables extended data length mode.<br>1: Enables extended data length mode.<br>For details, see **Section 16.5.8.2, Data length greater than 16 bits**. |
| 6 | CSIHnJE | Enables/disables job mode.<br>0: Disables job mode.<br>1: Enables job mode.<br>For details, see **Section 16.5.3.3, Job concept**.<br>The CSIHnCTL0.CSIHnJOBE, CSIHnTX0W.CSIHnEOJ, and CSIHnTX0W.CSIHnCIRE bits are enabled only when CSIHnJE = 1.<br>Setting this bit in slave mode is prohibited. |
| 5 | CSIHnDCS | Enables/disables data consistency check.<br>0: Disables data consistency check.<br>1: Enables data consistency check.<br>For details, see **Section 16.5.12.1, Data consistency check**. |
| 4 | CSIHnCSRI | Defines chip select signal behavior after last data transfer.<br>0: Chip select signal holds the active level.<br>1: Chip select signal returns to the inactive level.<br>Judgement of final data is interrupt output timing in Direct Access mode and FIFO mode. In Direct Access mode, CSIHnCTL1.CSIHnSLIT=1. |
| 3 | CSIHnLBM | Controls loop-back mode (LBM).<br>0: Deactivates loop-back mode.<br>1: Activates loop-back mode.<br>For details, see **Section 16.5.13, Loop-back mode**.<br>When LBM = 1:<br>- CSIHTSI is internally connected to CSIHTSO<br>- All CS are inactive (no influence of external world)<br>- Physical CSIHTSO pin is set to a fixed level (no influence of external world)<br>- External clock is not sent out |
| 2 | CSIHnSIT | Selects interrupt delay mode.<br>0: No delay is generated.<br>1: Half clock delay is generated for all interrupts.<br>This bit is only valid in master mode. In slave mode, no delay is generated.<br>For details, see **Section 16.4.2, General interrupt delay**.<br>Interrupts mean INT_CSIHTIC, INT_CSIHTIR, INT_CSIHTIRE and INT_CSIHTIJC. |
| 1 | CSIHnHSE | Enables/disables handshake mode.<br>0: Disables the handshake function.<br>1: Enables the handshake function.<br>For details see **Section 16.5.11, Handshake function**. |
| 0 | CSIHnSSE | Enables/disables the slave select function.<br>0: Input signal CSIHnSSIZ is ignored.<br>1: Input signal CSIHnSSIZ is recognized.<br>If the slave select function is not used, this bit must be set to 0 (see also **Section 16.5.2, Master/slave connections**). |

Details about CSIHnCTL1.CSIHnSSE are shown in the following tables.

**Table 16.15 Operation of the slave select function during reception**

| CSIHnCTL0.CSIHnRXE | CSIHnCTL1.CSIHnSSE | CSIHTSSIZ | Receive operation |
|---|---|---|---|
| 0 | — | — | Reception is disabled |
| 1 | 0 | — | Possible |
| 1 | 1 | 0 | Possible |
| 1 | 1 | 1 | Disabled |

**Table 16.16 Operation of the slave select function during transmission**

| CSIHnCTL0.CSIHnTXE | CSIHnCTL1.CSIHnSSE | CSIHTSSI | Transmit operation |
|---|---|---|---|
| 0 | — | — | Transmission is disabled |
| 1 | 0 | — | Possible |
| 1 | 1 | 0 | Possible |
| 1 | 1 | 1 | Disabled |

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

### 16.3.4 CSIHnCTL2 — CSIH control register 2

This register selects operating mode and the input clock for the four baud rate generators.

For details see **Section 16.5.5, Transmission clock selection**.

**Access:** This register can be read/write in 16-bit units.

**Address:** <CSIHn_base> + $0014_H$

**Value after reset:** $E000_H$ This register is initialized by reset operation.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnPRS[2:0] | | | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.17 CSIHnCTL2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 13 | CSIHnPRS[2:0] | These bits select the operation mode and the reference clock value. |
| 12 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

| CSIHnPRS2 | CSIHnPRS1 | CSIHnPRS0 | Selection of Reference Clock (PRSOUT) |
|---|---|---|---|
| 0 | 0 | 0 | CLKP_C (Master mode) |
| 0 | 0 | 1 | CLKP_C/2 (Master mode) |
| 0 | 1 | 0 | CLKP_C/4 (Master mode) |
| 0 | 1 | 1 | CLKP_C/8 (Master mode) |
| 1 | 0 | 0 | CLKP_C/16 (Master mode) |
| 1 | 0 | 1 | CLKP_C/32 (Master mode) |
| 1 | 1 | 0 | CLKP_C/64 (Master mode) |
| 1 | 1 | 1 | External clock via CSIHnSC(in) (Slave mode) |

In master mode, the following bits are used to set the transmission baud rate:

- CSIHnCTL2.CSIHnPRS[2:0]
- CSIHnCFGx.CSIHnBRSSx[1:0] (x = 0 to 7)
- CSIHnBRSy.CSIHnBRSy[11:0] (y = 0 to 3)

Four different baud rates can be pre-configured by the CSIHnBRSy registers (y = 0 to 3).

In CSIHnCFGx.CSIHnBRSSx[1:0] (x = 0 to 7) one of these baud rates can be selected for each chip select.

The following table shows the relationship between CSIHnCFGx.CSIHnBRSSx[1:0] and CSIHnBRSy.CSIHnBRSy[11:0] (y = 0 to 3) (x = 0 to 7).

| CSIHnCFGx.CSIHnBRSSx [1:0] (x = 0 to 7) | Baud rate setting bit to be selected |
|---|---|
| 00 | CSIHnBRS0.CSIHnBRS[11:0] |
| 01 | CSIHnBRS1.CSIHnBRS[11:0] |
| 10 | CSIHnBRS2.CSIHnBRS[11:0] |
| 11 | CSIHnBRS3.CSIHnBRS[11:0] |

The following table shows the relationship between the baud rate setting CSIHnBRSy [11:0] and the baud rate selected by the CSIHnBRSSx[1:0] bit when the bit value of the CSIHnPRS[2:0] bit is α.

| CSIHnBRSy[11:0](k) | Baud Rate |
|---|---|
| 0 | BRG stopped |
| 1 | CLKP_C / ($2^{\alpha}$ × 1 × 2) |
| 2 | CLKP_C / ($2^{\alpha}$ × 2 × 2) |
| 3 | CLKP_C / ($2^{\alpha}$ × 3 × 2) |
| 4 | CLKP_C / ($2^{\alpha}$ × 4 × 2) |
| ... | … |
| 4095 | CLKP_C / ($2^{\alpha}$ × 4095 × 2) |

When CLKP_C = 80MHz, Fast baud rate is 20.0Mbps (CLKP_C/4) in Master mode and 13.3Mbps (CLKP_C/6) in Slave mode.

The slowest baud rate is 152.6bps (CLKP_C/524160)

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

### 16.3.5 CSIHnMCTL1 — CSIH Memory control register 1

This register selects the conditions to generate the interrupt requests, CSIHTIC and CSIHTIR in FIFO mode.

**Access:** This register can be read/write in 32-bit units.

**Address:** <CSIHn_base> + $1000_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CSIHnFES[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CSIHnFFS[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.18 CSIHnMCTL1 register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22 to 16 | CSIHnFES[6:0] | Selects the conditions to generate the CSIHTIC interrupt (transmit data empty) in FIFO mode.<br>When the number of unsent data to be transmitted in FIFO (CSIHnSTR0. CSIHnSPF[7:0]) equals CSIHnMCTL1.CSIHnFES[6:0], the FIFO empty flag (CSIHnSTR0.CSIHnEMF bit) is set to 1, and the CSIHTIC interrupt request is generated. |
| 15 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 to 0 | CSIHnFFS[6:0] | Selects the conditions to generate the CSIHTIR interrupt (receive data full) in FIFO mode.<br>When the number of received data in the FIFO (CSIHnSTR0.CSIHnSRP[7:0]) equals CSIHnMCTL1.CSIHnFFS[6:0], the FIFO full flag (CSIHnSTR0.CSIHnFLF bit) is set to 1, and the CSIHTIR interrupt request is generated. |

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

### 16.3.6 CSIHnMCTL2 — CSIH Memory control register 2

This register controls the operation of the memory in dual buffer or transmit-only buffer mode and triggers to start communication in buffer mode.

**Access:** This register can be read/write in 32-bit units.

**Address:** <CSIHn_base> + $1004_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHn BTST | — | — | — | — | — | — | — | CSIHnND[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CSIHnSOP[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.19 CSIHnMCTL2 register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | CSIHnBTST | Provides a start trigger for buffer transfer.<br>0: No operation.<br>1: Issues the start transfer command.<br>The read value is always 0.<br>**CAUTION**<br>This bit can only be used in dual buffer mode and transmit-only buffer mode. |
| 30 to 24 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 23 to 16 | CSIHnND[7:0] | Specifies the number of data for each memory mode.<br>The read value indicates the number of remaining communication data. |

| CSIHn ND[7:0] | Dual buffer mode | Transmit-only buffer mode | FIFO mode | Direct access mode |
|---|---|---|---|---|
| $00_H$ | Send 0 data | Send 0 data | No influence | No influence |
| $01_H$ | Send 1 data | Send 1 data | No influence | No influence |
| ... | ... | ... | No influence | No influence |
| $3F_H$ | Send 63 data | Send 63 data | No influence | No influence |
| $40_H$ | Send 64 data | Send 64 data | No influence | No influence |
| ... | Prohibited | ... | No influence | No influence |
| $7F_H$ | Prohibited | Send 127 data | No influence | No influence |
| $80_H$ | Prohibited | Send 128 data | No influence | No influence |
| Other than the above | Setting is prohibited. | | | |

The values are automatically decremented after data transfer (Not decremented in direct access mode).

| | | |
|---|---|---|
| 15 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

**Table 16.19 CSIHnMCTL2 register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 to 0 | CSIHnSOP[6:0] | Selects the pointer of the data to be sent. |

| CSIHn SOP[6:0] | Dual buffer mode | Transmit-only buffer mode | FIFO mode | Direct access mode |
|---|---|---|---|---|
| $00_H$ | $0000_H$ | $0000_H$ | $0000_H$ | No influence |
| $01_H$ | $0004_H$ | $0004_H$ | $0004_H$ | No influence |
| ... | ... | ... | ... | No influence |
| $3F_H$ | $00FC_H$ | $00FC_H$ | $00FC_H$ | No influence |
| $40_H$ | Prohibited | $0100_H$ | $0100_H$ | No influence |
| ... | Prohibited | ... | ... | No influence |
| $7F_H$ | Prohibited | $01FC_H$ | $01FC_H$ | No influence |

**CAUTION**

If communication is forced to stop by setting CSIHnCTL0.PWR to 0 or CSIHnSTR0.CSIHnPCT to 1, these bits are cleared by hardware.
In FIFO mode, these bits indicate the send address.
In direct access mode, these bits are not incremented.

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

### 16.3.7 CSIHnTX0W — CSIH transmit data register 0 for word access

This register stores transmission data. In addition, it specifies the communication interrupt request, the end-of-job, the extended data length, and the chip select activation.

**Access:** This register can be read/write in 32-bit units.

**Address:** <CSIHn_base> + $1008_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnCIRE | CSIHnEOJ | CSIHnEDL | CSIHnSFN1 | CSIHnSFN0 | CSIHnISC | — | — | CSIHnCS7 | CSIHnCS6 | CSIHnCS5 | CSIHnCS4 | CSIHnCS3 | CSIHnCS2 | CSIHnCS1 | CSIHnCS0 |
| Value after reset | | | | 0 | 0 | 0 | 0 | 0 | | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnTX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | | | | | | | | | | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.20 CSIHnTX0W register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | CSIHnCIRE | Enables the communication interrupt request CSIHTIC in dual buffer or transmit-only buffer mode, or the job completion interrupt CSIHTIJC request in FIFO mode.<br>0: No interrupt is requested.<br>1: An interrupt is requested. Generates interrupt CSIHTIC or CSIHTIJC after transmission. For details, see **Section 16.4.3, CSIHTIC (communication interrupt)** and **Section 16.4.3.8, CSIHTIJC (job completion interrupt)**.<br>**CAUTION**<br>This bit is only valid when job mode is enabled (CSIHnCTL1.CSIHnJE = 1). |
| 30 | CSIHnEOJ | Specifies the end of a job.<br>0: Indicates that the job does not end. The job continues.<br>1: Indicates end-of-job data.<br>**CAUTION**<br>This bit is only valid when job mode is enabled (CSIHnCTL1.CSIHnJE = 1). This bit must be set to 0 in slave mode. |
| 29 | CSIHnEDL | Specifies whether the associated data requires the extended data length (EDL) option.<br>0: Normal operation<br>1: Enables the extended data length.<br>The associated data is transmitted as a 16-bit packet. No inter-data time or idle time will be inserted after the data is transmitted.<br>If CSIHnCTL1.CSIHnEDLE = 1 and CSIHnTX0W.CSIHnEDL = 1, the subsequent data must have the same CS selection. If CS is modified for the subsequent data, the correct operation is not assured.<br>**CAUTION**<br>This bit is only available if CSIHnCTL1.CSIHnEDLE = 1.<br>This bit is cleared when CSIHnCTL1.CSIHnEDLE is cleared. |

**Table 16.20 CSIHnTX0W register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 28, 27 | CSIHnSFN[1:0] | CSIHnSFN[1:0] will decide the number of the inserted dummy frames to extend idle time after the transmitted data.<br>00: No dummy frame will be inserted after the transmission.<br>01: Two dummy frames (32 bits) will be inserted after the transmission.<br>10: Four dummy frames (64 bits) will be inserted after the transmission.<br>11: Eight dummy frames (128 bits) will be inserted after the transmission.<br>**CAUTION**<br>This bit is only valid when silent mode is enabled (CSIHnCTL1.CSIHnSME = 1).<br>If the CSIHnEDL bit and one or both of the CSIHnSFN[1:0] bits are set, no silent communication will be added. |
| 26 | CSIHnISC | The idle state control bit decides whether the chip select signal change to inactive state after end of the transfer or not.<br>0: No change of the output level of CS signals after end of the transfer.<br>1: Change to inactive state of CS signals after end of the transfer.<br>**CAUTION**<br>This bit is only valid when flexible idle state mode is enabled (CSIHnCTL1.CSIHnISCE = 1). |
| 25, 24 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 23 to 16 | CSIHnCSx (x = 0 to 7) | Activates one or several chip select signals.<br>0: Activates chip select x for the associated transmission.<br>1: Deactivates chip select x for the associated transmission.<br>Setting CSIHnTX0W.CSIHnCSx = $FF_H$ is prohibited.<br>**CAUTION**<br>If several chip select signals are enabled for broadcasting, the configuration of one with CSIHnCFGx.CSIHnRCBx = 0 (dominant) is used. In this case, all dominant chip select signals must be set to precisely the same value.<br>In slave mode, set the CSIHnCSx bit to $FE_H$. |
| 15 to 0 | CSIHnTX[15:0] | Stores the transmission data. |

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

### 16.3.8 CSIHnTX0H — CSIH transmit data register 0 for half word access

This register stores the transmission data. This register is the same as bits 15 to 0 of register CSIHnTX0W.

The upper 16 bits of CSIHnTX0W are applied for transfer. Set transmit data to CSIHnTX0W before using this register because the value of CSIHnTX0W is undefined after the reset.

**Access:** This register can be read/write in 16-bit units.

**Address:** <CSIHn_base> + $100C_H$

**Value after reset:** Undefined

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnTX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | | | | | | | | | | | | | | | | |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.21 CSIHnTX0H register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | CSIHnTX[15:0] | Stores the transmission data. |

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

## 16.3.9 CSIHnRX0W — CSIH receive data register 0 for word access

This register stores the received data.

**Access:** This register can be read in 32-bit units.

**Address:** <CSIHn_base> + $1010_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CSIHn RPE | CSIHn TDCE | CSIHn CS7 | CSIHn CS6 | CSIHn CS5 | CSIHn CS4 | CSIHn CS3 | CSIHn CS2 | CSIHn CS1 | CSIHn CS0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnRX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | | | | | | | | | | | | | | | | |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.22 CSIHnRX0W register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is read. |
| 25 | CSIHnRPE | Indicates whether a reception data parity error was detected.<br>0: No parity error was detected on the associated reception data.<br>1: A parity error was detected on the associated reception data. |
| 24 | CSIHnTDCE | Indicates whether a transmission data consistency error was detected.<br>0: No consistency error was detected on the associated transmission data.<br>1: A consistency error was detected on the associated transmission data. |
| 23 to 16 | CSIHnCSx<br>(x = 7 to 0) | Indicates which chip select signal was activated.<br>0: Chip select x was activated for the associated reception.<br>1: Chip select x was deactivated for the associated reception. |
| 15 to 0 | CSIHnRX[15:0] | Stores the received data. |

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

### 16.3.10 CSIHnRX0H — CSIH receive data register 0 for half word access

This register stores the received data. This register is the same as bits 15 to 0 of register CSIHnRX0W.

**Access:** This register can be read in 16-bit units.
This register is lower 16-bit same as CSIHnRX0W register.

**Address:** <CSIHn_base> + $1014_H$

**Value after reset:** Undefined

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnRX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | | | | | | | | | | | | | | | | |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.23 CSIHnRX0H register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 15 to 0 | CSIHnRX[15:0] | Stores the received data. |

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

## 16.3.11 CSIHnMRWP0 — CSIH memory read/write pointer register 0

This register sets the pointers for reading from and writing to the dual or transmit-only buffer.

**Access:** This register can be read/written in 32-bit units.

**Address:** <CSIHn_base> + $1018_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CSIHnRRA[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CSIHnTRWA[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.24 CSIHnMRWP0 register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22 to 16 | CSIHnRRA[6:0] | Selects the read pointer of the buffer. |

| CSIHn RRA[6:0] | Dual buffer mode | Transmit-only buffer mode | FIFO mode | Direct access mode |
|---|---|---|---|---|
| $00_H$ | $0000_H$ | No influence | $0000_H$ | No influence |
| $01_H$ | $0004_H$ | No influence | $0004_H$ | No influence |
| ... | ... | No influence | ... | No influence |
| $3F_H$ | $00FC_H$ | No influence | $00FC_H$ | No influence |
| $40_H$ | Prohibited | No influence | $0100_H$ | No influence |
| ... | Prohibited | No influence | ... | No influence |
| $7F_H$ | Prohibited | No influence | $01FC_H$ | No influence |

These bits are automatically incremented when received data is read.
If an overrun error is generated while reading the CSIHnRX0W or CSIHnRX0H register, the read pointer is not incremented.
These bits are cleared when CSIHnSTCR0.CSIHnPCT is set to 1.
In direct access mode and transmit-only buffer mode, these bits are not incremented.
To perform write access in transmit-only mode, set $0000_H$ to these bits.
In FIFO mode, these bits indicate the read address of the received data.

| | | |
|---|---|---|
| 15 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

**Table 16.24 CSIHnMRWP0 register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 to 0 | CSIHnTRWA [6:0] | Selects the read/write pointer of the transmit buffer. |

| CSIHn TRWA[6:0] | Dual buffer mode | Transmit-only buffer mode | FIFO mode | Direct access mode |
|---|---|---|---|---|
| $00_H$ | $0000_H$ | $0000_H$ | $0000_H$ | No influence |
| $01_H$ | $0004_H$ | $0004_H$ | $0004_H$ | No influence |
| ... | ... | ... | ... | No influence |
| $3F_H$ | $00FC_H$ | $00FC_H$ | $00FC_H$ | No influence |
| $40_H$ | Prohibited | $0100_H$ | $0100_H$ | No influence |
| ... | Prohibited | ... | ... | No influence |
| $7F_H$ | Prohibited | $01FC_H$ | $01FC_H$ | No influence |

These bits are automatically incremented when the transmission data is written or read.
These bits are cleared when CSIHnSTCR0.CSIHnPCT is set to 1.
In direct access mode, these bits are not incremented.
In FIFO mode, these bits indicate the read/write address of transmission data.

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

## 16.3.12 CSIHnSTR0 — CSIH status register 0

This register indicates the status of CSIH.

**Access:** This register can be read in 32-bit units.

**Address:** <CSIHn_base> + $101C_H$

**Value after reset:** $0000\ 0010_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnSRP[7:0] | | | | | | | | CSIHnSPF[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHn TMOE | CSIHn OFE | — | — | — | — | — | — | CSIHn TSF | — | CSIHn FLF | CSIHn EMF | CSIHn DCE | — | CSIHn PE | CSIHn OVE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.25 CSIHnSTR0 register contents (1/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | CSIHnSRP[7:0] | Indicates the number of received data packets in FIFO mode.<br><br>**CSIHnSRP[7:0] / Description**<br>$00_H$ — Number of received data packets (0 to 128)<br>…<br>$80_H$<br>Other than the above — Reserve<br><br>These bits are cleared by CSIHnSTCR0.CSIHnPCT.<br>In direct access mode, dual buffer mode, or transmit-only buffer memory mode, this value is fixed to $00_H$.<br>In direct access mode, this bit is fixed to 0 because there is no pointer. In buffer mode, this bit is fixed to 0 because the number of data packets is managed by CSIHnMCTL2.CSIHnND[6:0]. |
| 23 to 16 | CSIHnSPF[7:0] | Indicates the number of unsent data in FIFO mode.<br>(The number of data written by the CPU/DMA to the CSIH is the number of data to be transmitted.)<br><br>**CSIHnSPF[7:0] / Description**<br>$00_H$ — Number of unsent data packets (0 to $128_D$)<br>…<br>$80_H$<br>Other than the above — Reserve<br><br>These bits are cleared by CSIHnSTCR0.CSIHnPCT.<br>In direct access mode, dual buffer mode, or transmit-only buffer memory mode, this value is fixed to $00_H$.<br>In direct access mode, this bit is fixed to 0 because there is no pointer. In buffer mode, this bit is fixed to 0 because the number of data packets is managed by CSIHnMCTL2.CSIHnND[6:0]. |

**Table 16.25 CSIHnSTR0 register contents (2/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | CSIHnTMOE | Time-out error flag in FIFO mode.<br>Indicates whether a time-out error was detected in FIFO mode.<br>0: No time out error is detected.<br>1: A time out error is detected.<br>For details, see **Section 16.5.12.3, Time-out error**.<br>This bit is cleared by CSIHnSTCR0.CSIHnTMOEC.<br>If this bit is set to 1 and cleared by CSIHnSTCR0.CSIHnTMOEC at the same time, setting to 1 is prioritized.<br>This bit is initialized when CSIHnCTL0.CSIHnPWR changes from 0 to 1 or from 1 to 0. |
| 14 | CSIHnOFE | Overflow error flag in FIFO mode.<br>Indicates whether an overflow error was detected in FIFO mode.<br>0: No overflow error is detected.<br>1: An overflow error is detected.<br>For details, see **Section 16.5.12.4, Overflow error**.<br>This bit is cleared by CSIHnSTCR0.CSIHnOFEC.<br>If this bit is set to 1 and cleared by CSIHnSTCR0.CSIHnOFEC at the same time, setting to 1 is prioritized.<br>This bit is initialized when CSIHnCTL0.CSIHnPWR changes from 0 to 1 or from 1 to 0.<br>When CSIHnCTL0.CSIHnPWR = 0, if transmission data (129) is written to CSIHnTX0W or CSIHnTX0H, an overflow error occurs. |
| 13 to 8 | Reserved | When read, the value after reset is read. |

**Table 16.25 CSIHnSTR0 register contents (3/4)**

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7</td><td>CSIHnTSF</td><td>Transfer status flag.<br>0: Idle state<br>1: Transmission is in progress or being prepared.<br>Conditions for setting and clearing this bit are shown in the following tables.
<table>
<tr><th rowspan="2">Master mode</th><th colspan="2">Set by</th><th rowspan="2">Cleared by</th></tr>
<tr><th>Direct access mode, FIFO mode</th><th>Double Buffer mode, transmit-only mode</th></tr>
<tr><td>Transmit-only mode</td><td rowspan="3">Data is written to a transmit register (CSIHnTX0W/ CSIHnTX0H)</td><td rowspan="3">Bit CSIHnMCTL2. CSIHnBTST is set</td><td rowspan="3">Within a half clock of the last serial clock edge*1</td></tr>
<tr><td>Transmit/receive mode</td></tr>
<tr><td>Receive-only mode</td></tr>
</table>
Note 1. TSF flag will only be cleared if data can be transferred from shift register to empty RX register. If RX register is not empty, TSF flag will remain high until RX register has been read.
<table>
<tr><th rowspan="2">Slave mode</th><th colspan="2">Set by</th><th rowspan="2">Cleared by</th></tr>
<tr><th>Direct access mode, FIFO mode</th><th>Double Buffer mode, transmit-only mode</th></tr>
<tr><td>Transmit-only mode</td><td rowspan="2">Data is written to a transmit register (CSIHnTX0W/ CSIHnTX0H)</td><td rowspan="2">Bit CSIHnMCTL2. CSIHnBTST is set</td><td rowspan="2">Within a half clock of the last serial clock edge*2</td></tr>
<tr><td>Transmit/receive mode</td></tr>
<tr><td>Receive-only mode</td><td>Input timing of CSIHnSC</td><td></td><td></td></tr>
</table>
Note 2. TSF flag will only be cleared if data can be transferred from shift register to empty RX register. If RX register is not empty, TSF flag will remain high until RX register has been read.<br>This bit is cleared by CSIHnSTCR0.CSIHnPCT.<br>CAUTION<br>The TSF flag will be set with a certain delay after the set event described in the tables above has occured (e.g. “Data is written to a transmit register”). Consider this behaviour when reading the status of this bit. Alternatively use the corresponding interrupt / interrupt status flag to monitor the transfer status.</td></tr>
<tr><td>6</td><td>Reserved</td><td>When read, the value after reset is read.</td></tr>
<tr><td>5</td><td>CSIHnFLF</td><td>A flag indicating that the buffer is full in FIFO mode.<br>0: FIFO buffer is not full.<br>1: FIFO buffer is full.<br>This bit is cleared by CSIHnSTCR0.CSIHnPCT.<br>The FIFO buffer might be filled with unsent data or received data.<br>Set condition:<br>– If CSIHnSTR0.CSIHnSRP[7:0] plus CSIHnSTR0.CSIHnSPF[7:0] equals 80H.<br>Clear condition:<br>– If CSIHnSTR0.CSIHnSRP[7:0] plus CSIHnSTR0.CSIHnSPF[7:0] not equals 80H.</td></tr>
</table>

**Table 16.25 CSIHnSTR0 register contents (4/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 | CSIHnEMF | A flag indicating that the buffer is empty in FIFO mode.<br>0: FIFO buffer is not empty.<br>1: FIFO buffer is empty.<br>This bit is set to 1 by CSIHnSTCR0.CSIHnPCT.<br>This bit is set to 1 when the setting value of CSIHnMCTL1.CSIHnFES[6:0] and the value for the CSIHnSTR0.CSIHnSPF[7:0] bit match.<br>The FIFO buffer might be filled with unsent data or received data.<br>Set condition:<br>– If CSIHnSTR0.CSIHnSRP[7:0] plus CSIHnSTR0.CSIHnSPF[7:0] equals 00H, This bit is set by CSIHnSTCR0.CSIHnPCT bit.<br>Clear condition:<br>– If CSIHnSTR0.CSIHnSRP[7:0] plus CSIHnSTR0.CSIHnSPF[7:0] not equals 00H. |
| 3 | CSIHnDCE | Data consistency checks error flag.<br>0: No data consistency error is detected.<br>1: Data consistency error is detected.<br>This bit is cleared by writing 1 to CSIHnSTCR0.CSIHnDCEC.<br>If this bit is set to 1 and cleared by CSIHnSTCR0.CSIHnDCEC at the same time, setting to 1 is prioritized.<br>This bit is initialized when CSIHnCTL0.CSIHnPWR changes from 0 to 1 or from 1 to 0. |
| 2 | Reserved | When read, the value after reset is read. |
| 1 | CSIHnPE | Parity error flag<br>0: No parity error is detected.<br>1: Parity error is detected.<br>This bit is cleared by writing 1 to CSIHnSTCR0.CSIHnPEC.<br>Write access to this bit is enabled when CSIHnCTL0.CSIHnPWR = 0. This bit is initialized when CSIHnCTL0.CSIHnPWR changes from 0 to 1 or from 1 to 0.<br>When this bit is set to 1 by the detection of parity error and cleared to 0 by CSIHnSTCR0.CSIHnPEC simultaneously, setting to 1 is prioritized. |
| 0 | CSIHnOVE | Overrun error flag (Fixed to 0 in dual buffer mode).<br>0: No overrun error is detected.<br>1: Overrun error is detected.<br>This bit is cleared by writing 1 to CSIHnSTCR0.CSIHnOVEC. When this bit is set to 1 by the detection of overrun error and cleared to 0 by CSIHnSTCR0.CSIHnOVEC simultaneously, setting to 1 is prioritized.<br>This bit is Initialized when CSIHnCTL0.CSIHnPWR changes from 0 to 1 or from 1 to 0. |

**Table 16.26 Behavior in memory mode**

| | | Mode | | | |
|---|---|---|---|---|---|
| **Bit name** | **Bit position** | **Direct access** | **FIFO** | **Transmit-only** | **Dual buffer** |
| CSIHnSRP[7:0] | 31 to 24 | Fixed to 0 | Number of received words | Fixed to 0 | Fixed to 0 |
| CSIHnSPF[7:0] | 23 to 16 | Fixed to 0 | Number of unsent data | Fixed to 0 | Fixed to 0 |
| CSIHnTMOE | 15 | Fixed to 0 | Time-out error flag | Fixed to 0 | Fixed to 0 |
| CSIHnOFE | 14 | Fixed to 0 | Overflow error flag | Fixed to 0 | Fixed to 0 |
| CSIHnTSF | 7 | 0: Idle state<br>1: Transmission is in progress or being prepared*1 | | | |
| CSIHnFLF | 5 | Fixed to 0 | FIFO full flag | Fixed to 0 | Fixed to 0 |
| CSIHnEMF | 4 | Fixed to 1 | FIFO empty flag | Fixed to 1 | Fixed to 1 |

**Table 16.26 Behavior in memory mode**

| Bit name | Bit position | Mode | | | |
|---|---|---|---|---|---|
| | | Direct access | FIFO | Transmit-only | Dual buffer |
| CSIHnDCE | 3 | 0: No error is detected.<br>1: An error is detected. | | | |
| CSIHnPE | 1 | 0: No error is detected.<br>1: An error is detected. | | | |
| CSIHnOVE | 0 | 0: No error is detected.<br>1: An error is detected. | 0: No error is detected.<br>1: An error is detected. | 0: No error is detected.<br>1: An error is detected. | Fixed to 0 |

Note 1. TSF flag will only be cleared if data can be transferred from shift register to empty RX register. If RX register is not empty, TSF flag will remain high until RX register has been read.

**CAUTIONS**

1. For the setting of this register, see **Table 16.39, Notes on Setting Registers**.
2. The TSF flag will be set with a certain delay after the set event (e.g. "Data is written to a transmit register"). Consider this behaviour when reading the status of this bit. Alternatively use the corresponding interrupt / interrupt status flag to monitor the transfer status.

## 16.3.13 CSIHnSTCR0 — CSIH status clear register 0

This register clears the status flags of the CSIHnSTR0 status register.

**Access:** This register can be read/written in 16-bit units.
When read, the value $0000_H$ is always returned.

**Address:** <CSIHn_base> + $1020_H$

**Value after reset:** $0000_H$ This register is initialized by reset operation.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHn TMOEC | CSIHn OFEC | — | — | — | — | — | CSIHn PCT | — | — | — | — | CSIHn DCEC | — | CSIHn PEC | CSIHn OVEC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R/W | R | R | R | R | R/W | R | R/W | R/W |

**Table 16.27 CSIHnSTCR0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | CSIHnTMOEC | Controls the time-out error flag clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the time out error flag (CSIHnSTR0.CSIHnTMOE). |
| 14 | CSIHnOFEC | Controls the overflow error flag clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the overflow error flag (CSIHnSTR0.CSIHnOFE). |
| 13 to 9 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 8 | CSIHnPCT | Controls the FIFO pointer clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the FIFO buffer pointers below (in FIFO mode and dual buffer mode only) and the status bits. |
| 7 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 | CSIHnDCEC | Controls the data consistency error flag clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the data consistency error flag (CSIHnSTR0.CSIHnDCE). |
| 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | CSIHnPEC | Controls the parity error flag clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the parity error flag (CSIHnSTR0.CSIHnPE). |
| 0 | CSIHnOVEC | Controls the overrun error flag clear command.<br>0: No operation. The read value is always 0.<br>1: Clears the overrun error flag (CSIHnSTR0.CSIHnOVE). |

| FIFO buffer pointer | Status bit |
|---|---|
| CSIHnMRWPO.CSIHnTRWA[6:0]<br>CSIHnMRWPO.CSIHnRRA[6:0]<br>CSIHnMCTL2.CSIHnSOP[6:0] | CSIHnSTR0.CSIHnSPF[7:0]<br>CSIHnSTR0.CSIHnSRP[7:0]<br>CSIHnSTR0.CSIHnFLF<br>CSIHnSTR0.CSIHnTSF |

Additionally, the CSIHnSTR0.CSIHnEMF bit is set to 1 (FIFO empty) (in FIFO mode only).

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

## 16.3.14 CSIHnMCTL0 — CSIH Memory control register 0

This register selects the memory mode and the time-out setting.

**Access:** This register can be read/written in 16-bit units.

**Address:** <CSIHn_base> + $1040_H$

**Value after reset:** $001F_H$ This register is initialized by reset operation.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CSIHn MMS[1:0] | | — | — | — | CSIHnTO[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 16.28 CSIHnMCTL0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 10 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 9 to 8 | CSIHnMMS [1:0] | Selects the memory mode.<br>**CSIHn MMS1 / CSIHn MMS0 / Description**<br>0 / 0 / FIFO mode<br>0 / 1 / Dual buffer mode<br>1 / 0 / Transmit-only buffer mode<br>1 / 1 / Prohibited<br>After a change of the memory mode, the respective buffer pointers must be cleared by setting the CSIHnSTCR0.CSIHnPCT bit to 1.<br>In direct access mode, the setting of these bits is ignored. |
| 7 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 to 0 | CSIHnTO[4:0] | Selects the time-out setting in FIFO mode.<br>**CSIHnTO[4:0] / Description**<br>$00000_B$ / No time-out is detected<br>$00001_B$ / Time-out is (1 × 8 × BRG output clocks)<br>$00010_B$ / Time-out is (2 × 8 × BRG output clocks)<br>...<br>$11111_B$ / Time-out is (31 × 8 × BRG output clocks)<br>**CAUTION**<br>Changing the time-out setting is only permitted when CSIHnCTL0.CSIHnPWR = 0.<br>Set the CSIHnTO[4:0] bit to $00000_B$ in direct access mode, dual buffer mode, or transmit-only mode (except FIFO mode).<br>For details about time-out detection, see also **Section 16.5.12.3, Time-out error**. |

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

## 16.3.15 CSIHnCFGx — CSIH Configuration register (x = 0 to 7)

These eight registers specify for each chip select signal CSIHTCSSx used baud rate generator, parity, data length, recessive configuration for broadcasting, serial data direction, clock and data phase, idle enforcement configuration, idle time, hold time, inter-data time and setup time.

### Slave mode

In slave mode, the transmission protocol setting of the CSIHnCFG0 register is effective.

- CSIHnPS0[1:0]: parity usage
- CSIHnDLS0[3:0]: data length selection
- CSIHnDIR0: data direction
- CSIHnCKP0, CSIHnDAP0: clock and data phase

In slave mode, set 0 to all the bits other than above in the CSIHnCFG0 register, and the CSIHnCFG1 to CSIHnCFG7 registers.

**Access:** This register can be read/written in 32-bit units.

**Address:** CSIHnCFG0: <CSIHn_base> + $1044_H$
CSIHnCFG1: <CSIHn_base> + $1048_H$
CSIHnCFG2: <CSIHn_base> + $104C_H$
CSIHnCFG3: <CSIHn_base> + $1050_H$
CSIHnCFG4: <CSIHn_base> + $1054_H$
CSIHnCFG5: <CSIHn_base> + $1058_H$
CSIHnCFG6: <CSIHn_base> + $105C_H$
CSIHnCFG7: <CSIHn_base> + $1060_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHnBRSSx [1:0] | | CSIHnPSx[1:0] | | CSIHnDLSx[3:0] | | | | — | — | — | — | CSIHn RCBx | CSIHn DIRx | CSIHn CKPx | CSIHn DAPx |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSIHn IDLx | CSIHnIDx[2:0] | | | CSIHnHDx[3:0] | | | | CSIHnINx[3:0] | | | | CSIHnSPx[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.29 CSIHnCFGx register contents (1/5)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | CSIHnBRSSx [1:0] | A register used to set the baud rate. |

| CSIHn BRSSx1 | CSIHn BRSSx0 | Baud rate setting register selection |
|---|---|---|
| 0 | 0 | The baud rate is set according to the CSIHnBRS0 setting |
| 0 | 1 | The baud rate is set according to the CSIHnBRS1 setting |
| 1 | 0 | The baud rate is set according to the CSIHnBRS2 setting |
| 1 | 1 | The baud rate is set according to the CSIHnBRS3 setting |

The maximum value for setting the baud rate, combining the CSIHnCTL2.CSIHnPRS to 0 setting, must be as follows:
Master mode: CLKP_C/4 (Up to 20 MHz bit/s)
Slave mode: CLKP_C/6 (Up to 20 MHz bit/s)

**Table 16.29 CSIHnCFGx register contents (2/5)**

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>29, 28</td><td>CSIHnPSx[1:0]</td><td>Selects the parity for sending or receiving chip select signal x.
<table>
<tr><th>CSIHn PSx1</th><th>CSIHn PSx0</th><th>Transmission</th><th>Reception</th></tr>
<tr><td>0</td><td>0</td><td>No parity is transmitted</td><td>No parity is expected</td></tr>
<tr><td>0</td><td>1</td><td>Adds parity bit fixed to 0</td><td>Parity bit is expected but not judged</td></tr>
<tr><td>1</td><td>0</td><td>Adds odd parity only</td><td>Odd parity bit is expected</td></tr>
<tr><td>1</td><td>1</td><td>Adds even parity</td><td>Even parity bit is expected</td></tr>
</table></td></tr>
<tr><td>27 to 24</td><td>CSIHnDLSx [3:0]</td><td>Selects the data length for chip select sinal x.
<table>
<tr><th>CSHBAnDLSx[3:0]</th><th>Data length</th></tr>
<tr><td>$0000_B$</td><td>16 bits</td></tr>
<tr><td>$0001_B$</td><td>1 bit</td></tr>
<tr><td>$0010_B$</td><td>2 bits</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>$1111_B$</td><td>15 bits</td></tr>
</table>
CAUTION<br>Data length between 1 bit and 6 bits requires that the EDL function is used (see also Section 16.5.8.2, Data length greater than 16 bits). When CSIHnTX0W.CSIHnEDL = 1, the setting of this bit has no effect.<br>When CSIHnTX0W.CSIHnEDL = 0 (the data length is 16 bits), the setting of this bit is valid. Only when the previous transmit data is 16 bits while CSIHnEDL = 1, writing 1 to this bit is enabled.</td></tr>
<tr><td>23 to 20</td><td>Reserved</td><td>When read, the value after reset is read. When writing, write the value after reset.</td></tr>
<tr><td>19</td><td>CSIHnRCBx</td><td>Selects the recessive configuration for broadcasting for chip select x:<br>0: Dominant (higher priority)<br>1: Recessive (lower priority)<br>For details, see Section 16.5.3.1, Configuration registers</td></tr>
<tr><td>18</td><td>CSIHnDIRx</td><td>Selects the serial data direction of chip select signal x.<br>0: Data is transmitted/received with MSB first.<br>1: Data is transmitted/received with LSB first.<br>For details, see Section 16.5.9, Serial data direction selection.</td></tr>
</table>

**Table 16.29 CSIHnCFGx register contents (3/5)**

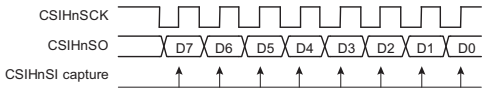
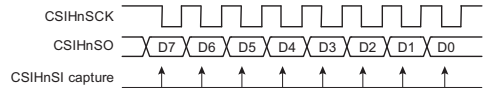
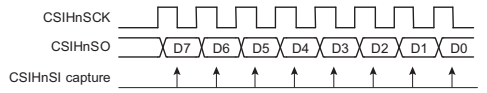
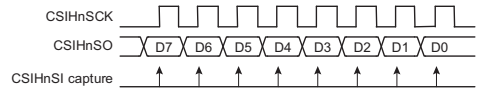
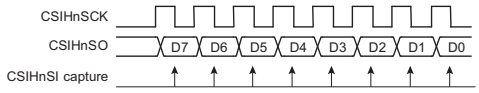
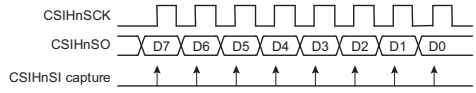
<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>17</td><td>CSIHnCKPx</td><td rowspan="2">• CSIHnCTL1.CSIHnCKR = 0<br>• CSIHnCTL1.CSIHnSLRS = 0<br><table><tr><th>CSIHn CKPx</th><th>CSIHn DAPx</th><th>Clock and data phase selection</th></tr><tr><td>0</td><td>0</td><td> 
</td></tr><tr><td>0</td><td>1</td><td> 
</td></tr><tr><td>1</td><td>0</td><td> 
</td></tr><tr><td>1</td><td>1</td><td> 
</td></tr></table>
• CSIHnCTL1.CSIHnCKR = 1<br>• CSIHnCTL1.CSIHnSLRS = 0<br><table><tr><th>CSIHn CKPx</th><th>CSIHn DAPx</th><th>Clock and data phase selection</th></tr><tr><td>0</td><td>0</td><td> 
</td></tr><tr><td>0</td><td>1</td><td> 
</td></tr><tr><td>1</td><td>Don’t care</td><td>Prohibition</td></tr></table></td></tr>
<tr><td>16</td><td>CSIHnDAPx</td></tr>
</table>

**Table 16.29 CSIHnCFGx register contents (4/5)**

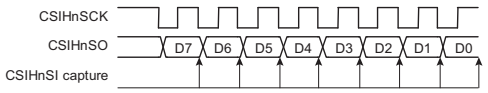
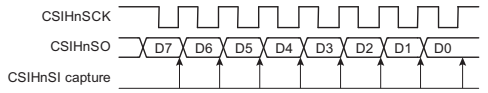
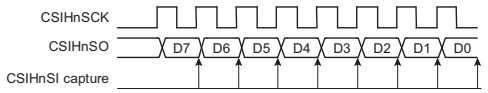
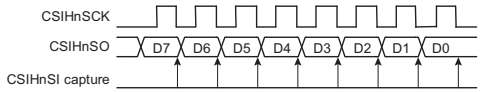
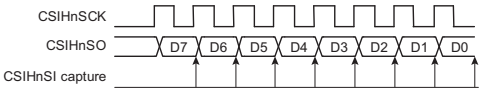
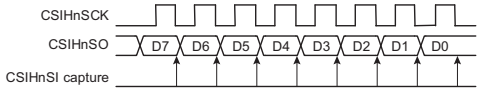
<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>17</td><td>CSIHnCKPx</td><td rowspan="2">• CSIHnCTL1.CSIHnCKR = 0<br>• CSIHnCTL1.CSIHnSLRS = 1<br><table><tr><th>CSIHn CKPx</th><th>CSIHn DAPx</th><th>Clock and data phase selection</th></tr><tr><td>0</td><td>0</td><td>  </td></tr><tr><td>0</td><td>1</td><td>  </td></tr><tr><td>1</td><td>0</td><td>  </td></tr><tr><td>1</td><td>1</td><td>  </td></tr></table><br>• CSIHnCTL1.CSIHnCKR = 1<br>• CSIHnCTL1.CSIHnSLRS = 1<br><table><tr><th>CSIHn CKPx</th><th>CSIHn DAPx</th><th>Clock and data phase selection</th></tr><tr><td>0</td><td>0</td><td>  </td></tr><tr><td>0</td><td>1</td><td>  </td></tr><tr><td>1</td><td>Don’t care</td><td>Prohibition</td></tr></table></td></tr>
<tr><td>16</td><td>CSIHnDAPx</td></tr>
<tr><td>15</td><td>CSIHnIDLx</td><td>Selects the idle enforcement configuration for chip select x:<br>0: If the CSIHnTX0W.CSIHnCSx settings of two consecutive transmissions are different, all of the CSIHTCSS0 to 7 is once inactivated between the two transmissions. If the CSIHnTX0W.CSIHnCSx settings of two consecutive transmissions are same, there is no inactive period between the two transmissions.<br>1: Regardless of CSIHnTX0W.CSIHnCSx settings of two consecutive transmissions, all of the CSIHTCSS0 to 7 is once inactivated between the two transmissions.<br>This bit is only available in master mode.<br>When CSIHnCTL1.CSIHnJE = 1 and CSIHnTX0W.CSIHnEOJ = 1, even if CSIHnCFG0 to 7.CSIHnIDLx = 0, idle state is surely insert.<br>For details about the idle state, see Section 16.5.3, Chip selection (CS) features.</td></tr>
</table>

**Table 16.29 CSIHnCFGx register contents (5/5)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 14 to 12 | CSIHnIDx[2:0] | Selects the idle time for chip select signal x.<br>**CSIHnIDx[2:0]** / **Idle time**<br>$000_B$ / 0.5 transmission clock cycle<br>$001_B$ / 1 transmission clock cycle<br>$010_B$ / 1.5 transmission clock cycle<br>... / ... (2.5, 3.5, 4.5, 6.5)<br>$111_B$ / 8.5 transmission clock cycle<br>These bits are only available in master mode. |
| 11 to 8 | CSIHnHDx[3:0] | Specifies the hold time for chip select signal x in transmission clock cycles.<br>**CSIHnHDx[3:0]** / **Hold time with CSIHnCTL1.CSIHnSIT = 0** / **Hold time with CSIHnCTL1.CSIHnSIT = 1**<br>$0000_B$ / 0.5 transmission clock cycle / 1 transmission clock cycle<br>$0001_B$ / 1 transmission clock cycle / 1.5 transmission clock cycle<br>$0010_B$ / 1.5 transmission clock cycle / 2 transmission clock cycle<br>... / ... (2.5, 3.5, 4.5, 6.5, 8.5, 9.5 10.5, 11.5, 12.5, 14.5, 16.5, 18.5) / ... (3.0, 4.0, 5.0, 7.0, 9.0, 10.0, 11.0, 12.0, 13.0, 15.0, 17.0, 19.0)<br>$1111_B$ / 20.5 transmission clock cycle / 21 transmission clock cycle<br>These bits are only available in master mode. |
| 7 to 4 | CSIHnINx[3:0] | Specifies the inter-data delay time for chip select signal x in transmission clock cycles.<br>**CSIHnINx[3:0]** / **Inter-data delay with CSIHnCTL1.CSIHnSIT = 0** / **Inter-data delay with CSIHnCTL1.CSIHnSIT = 1**<br>$0000_B$ / 0.0 transmission clock cycle / 0.5 transmission clock cycle<br>$0001_B$ / 0.5 transmission clock cycle / 1.0 transmission clock cycle<br>$0010_B$ / 1.0 transmission clock cycle / 1.5 transmission clock cycle<br>$0011_B$ / 2.0 transmission clock cycle / 2.5 transmission clock cycle<br>... / ... (3.0, 4.0, 6.0, 8.0, 9.0, 10.0, 11.0, 12.0, 14.0, 16.0, 18.0) / ... (3.5, 4.5, 6.5, 8.5, 9.5, 10.5, 11.5 12.5, 14.5, 16.5, 18.5)<br>$1111_B$ / 20.0 transmission clock cycle / 20.5 transmission clock cycle<br>These bits are only available in master mode. |
| 3 to 0 | CSIHnSPx[3:0] | Specifies the setup time for chip select signal x in transmission clock cycles.<br>**CSIHnSPx[3:0]** / **Setup time**<br>$0000_B$ / 0.5 transmission clock cycle<br>$0001_B$ / 1 transmission clock cycle<br>$0010_B$ / 1.5 transmission clock cycle<br>... / ... (2.5, 3.5, 4.5, 6.5, 8.5, 9.5 10.5, 11.5, 12.5, 14.5, 16.5, 18.5)<br>$1111_B$ / 20.5 transmission clock cycle<br>These bits are only available in master mode. |

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

### 16.3.16 CSIHnBRSx — CSIH baud rate setting register (x = 0 to 3)

This register sets the baud rate for each chip select signal.

With CSIHnCFGx.CSIHnBRSSx[1:0] bits, one of the four types of baud rate settings can be selected for each chip select signal. (x = 0 to 7)

For details of baud rate setting, see **Section 16.5.5, Transmission clock selection**.

**Access:** This register can be read/write in 16-bit units.

**Address:** CSIHnBRS0: <CSIHn_base> + $1068_H$
CSIHnBRS1: <CSIHn_base> + $106C_H$
CSIHnBRS2: <CSIHn_base> + $1070_H$
CSIHnBRS3: <CSIHn_base> + $1074_H$

**Value after reset:** $0000_H$ This register is initialized by reset operation.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | CSIHnBRSx[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.30 CSIHnBRSx register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 15 to 12 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 11 to 0 | CSIHnBRSx [11:0] | 0: BRG stopped<br>1: CLKP_C / ($2^{\alpha}$ × 1 × 2)<br>2: CLKP_C / ($2^{\alpha}$ × 2 × 2)<br>3: CLKP_C / ($2^{\alpha}$ × 3 × 2)<br>4: CLKP_C / ($2^{\alpha}$ × 4 × 2)<br>.<br>.<br>.<br>4095: CLKP_C / ($2^{\alpha}$ × 4095 × 2)<br><br>α is the value of CSIHnCTL2.CSIHnPRS (bit 2 to 0). |

**CAUTION**

For the setting of this register, see **Table 16.39, Notes on Setting Registers**.

## 16.3.17 PMMAnCTL — PMM control register

This register controls the operation clock.

**Access:** This register can be read/write in 32-bit units.

**Address:** <CSIHn_base> + $2000_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMAnPWR | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.31 PMMAnCTL register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 31 | PMMAnPWR | Controls the operation clock:<br>0: Stops operation clock<br>1: Provides operation clock<br>Clearing PMMAnPWR to 0 resets the internal circuits (internal registers will have reset value after enabling), stops operation. No clock is provided to internal circuits.<br>**NOTE**<br>PMMAnTCNT, PMMAnSTR0/1, and PMMAnRX registers are cleared.Others aren't cleared.<br>If PMMAnPWR is cleared during TG handling all TG's are immediately aborted. |
| 30 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

## 16.3.18 PMMAnTCTLx — PMM TG control register (x = 0 to 7)

This register controls the operation of priority management module.

**Access:** This register can be read/write in 32-bit units.

**Address:** PMMAnTCTL0: <CSIHn_base> + $2004_H$
PMMAnTCTL1: <CSIHn_base> + $2008_H$
PMMAnTCTL2: <CSIHn_base> + $200C_H$
PMMAnTCTL3: <CSIHn_base> + $2010_H$
PMMAnTCTL4: <CSIHn_base> + $2014_H$
PMMAnTCTL5: <CSIHn_base> + $2018_H$
PMMAnTCTL6: <CSIHn_base> + $201C_H$
PMMAnTCTL7: <CSIHn_base> + $2020_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMAnSTx | PMMAnSPx | — | — | — | — | — | — | PMMAnTGLGx[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | PMMAnPRIOx[2:0] | | | PMMAnMD | PMMAnTOx[1:0] | | PMMAnSWTx | — | PMMAnHWTx[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | W | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.32 PMMAnTCTLx register contents (1/2)**

| Bit position | Bit name | Function |
|---|---|---|
| 31 | PMMAnSTx | Start trigger to enable the TG<br>0: No effect<br>1: Start trigger to enable the TG<br>By setting the PMMAnSTx bit the PMMAnTTXm register will be cleared to avoid the use of old data.<br>This bit is always read as 0 |
| 30 | PMMAnSPx | Stop trigger to disable the TG<br>0: No effect<br>1: Stop trigger to disable the TG<br>At the time of Enable and Pending mode, by setting this bit TGx will be set to Disabled mode immediately. Only if the TGx is currently in Active or Wait mode, the current Job will be finished before changing the mode. After disabling a TG it is not possible to resume it automatically.<br>For details on each state, see **Section 16.7.1.2, TG status.**<br>This bit is always read as 0<br>Writing to the PMMAnSPx bit is enabled in the mode other than the Disable mode<br>When the PMMAnSPx bit is set to 1 in Disable mode, status flags are not updated and the operation remains in Disabled mode.<br>This bit is disabled after the read data reception finish at the time of active wait. |
| 29 to 24 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 23 to 16 | PMMAnTGLGx [7:0] | Specify the number of data packages (32bit (data + control bits)) for TGx.<br>If the frame length is 256, "$00_H$" should be set.<br>Max. TG length: 256 |
| 15, 14 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 13 to 11 | PMMAnPRIOx [2:0] | Specify the priority of the TG<br>$000_B$: Highest priority<br>:<br>$101_B$: lowest priority<br>Other than above: Setting prohibited |

**Table 16.32 PMMAnTCTLx register contents (2/2)**

| Bit position | Bit name | Function |
|---|---|---|
| 10 | PMMAnMD | Specifies the transfer mode<br>0: Transmit only mode (Tx)<br>1: Transmit and receive mode (Tx/Rx) |
| 9, 8 | PMMAnTOx [1:0] | Specify the trigger option<br>$00_B$: Trigger disable<br>$01_B$: Software trigger<br>$10_B$: Hardware trigger<br>$11_B$: Hardware and Software trigger<br>When the PMMAnTOx[1:0] bits are set to 00, even if the bit for the software or hardware trigger is set to 1, it is ignored. |
| 7 | PMMAnSWTx | Software Trigger to set TGx to pending mode<br>This bit is valid in Enable mode. When 1 is written to the PMMAnST bit and this bit simultaneously in Disable mode, this bit is also valid.<br>This bit is always read as 0<br>Writing to the PMMAnSWTx bit is enabled when the TG is in Disable or Enable mode<br>If the PMMAnSWTx bit is only set to 1, this setting is ignored and the operation remains in Disable mode. |
| 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 to 0 | PMMAnHWTx [5:0] | Select the hardware trigger source<br>Setting 100011 or more number to PMMAnTCTL.PMMAnHWTm[5:0] is prohibited.<br>Details about hardware trigger sources, see **Table 16.53**. |

**NOTES**

1. Writing to other than the PMMAnSPx and PMMAnSWTx bits in the PMMAnTCTLx register is only allowed when the TG is in Disable mode.
2. When the PMMAnSTx and PMMAnSWTx bits are set to 1 simultaneously while the TG is in Disable mode, transition to the Pending mode is made.
3. If the PMMAnSTx and PMMAnSPx bits are input simultaneously, the PMMAnSPx bit is ignored because the PMMAnSPx bit has priority

## 16.3.19 PMMAnTTXm — PMM TG transmit buffer (x = 0 to 7)

This register is transmitting buffer.

**Access:** This register can be read/write in 32-bit units. (Writing in from CPU to this register is prohibited.)

**Address:**
PMMAnTTX0: <CSIHn_base> + $2024_H$
PMMAnTTX1: <CSIHn_base> + $2028_H$
PMMAnTTX2: <CSIHn_base> + $202C_H$
PMMAnTTX3: <CSIHn_base> + $2030_H$
PMMAnTTX4: <CSIHn_base> + $2034_H$
PMMAnTTX5: <CSIHn_base> + $2038_H$
PMMAnTTX6: <CSIHn_base> + $203C_H$
PMMAnTTX7: <CSIHn_base> + $2040_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMAnTTXm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMAnTTXm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 16.33 PMMAnTTXm register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 31 to 0 | PMMAnTTXm | Data buffer of TGx for the transmit path. |

## 16.3.20 PMMAnRX — PMM TG receive buffer

This register is receiving buffer.

**Access:** This register can be read in 32-bit units.

**Address:** PMMAnRX: <CSIHn_base> + $2044_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMAnRX[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMAnRX[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.34 PMMAnRX register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 31 to 0 | PMMAnRX | Data buffer for the receive path<br>In transmit-only mode, data is not stored in the PMMAnRX register. |

## 16.3.21 PMMAnTCNTx — PMM TGx counter (x = 0 to 7)

This register is counter which indicated the remaining data transfers to PMMA for TGx.

**Access:** This register can be read in 32-bit units.

**Address:** PMMAnTCNT0: <CSIHn_base> + $2048_H$
PMMAnTCNT1: <CSIHn_base> + $204C_H$
PMMAnTCNT2: <CSIHn_base> + $2050_H$
PMMAnTCNT3: <CSIHn_base> + $2054_H$
PMMAnTCNT4: <CSIHn_base> + $2058_H$
PMMAnTCNT5: <CSIHn_base> + $205C_H$
PMMAnTCNT6: <CSIHn_base> + $2060_H$
PMMAnTCNT7: <CSIHn_base> + $2064_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | PMMAnTCNTx[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.35 PMMAnTCNTx register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is read. |
| 7 to 0 | PMMAn TCNTx[7:0] | Counter which indicated the remaining data transfers to PMMA for TGx<br>Will be loaded with value of PMMAnTGLGx[7:0] when PMMAnCTLx.PMMAnSTx bit is set.<br>Every time data is forwarded to CSIHnTX0W register, this register value does -1. |

## 16.3.22 PMMAnSTR0 — PMM TG status register 0

This register indicates the status of TG.

**Access:** This register can be read in 32-bit units.

**Address:** <CSIHn_base> + $2068_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMAnEN7 | PMMAnEN6 | PMMAnEN5 | PMMAnEN4 | PMMAnEN3 | PMMAnEN2 | PMMAnEN1 | PMMAnEN0 | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMAnAF7 | PMMAnAF6 | PMMAnAF5 | PMMAnAF4 | PMMAnAF3 | PMMAnAF2 | PMMAnAF1 | PMMAnPF0 | PMMAnPF7 | PMMAnPF6 | PMMAnPF5 | PMMAnPF4 | PMMAnPF3 | PMMAnPF2 | PMMAnPF1 | PMMAnPF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.36 PMMAnSTR0 register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 31 to 24 | PMMAnENx | Indicates the mode disabled enabled<br>0: TGx is disabled<br>1: TGx is enabled<br>For detail, see **Section 16.7.1.2, TG status**. |
| 23 to 16 | Reserved | When read, the value after reset is read. |
| 15 to 8 | PMMAnAFx | Indicates if TGx is set to status active<br>0: TGx is not in status active.<br>1: TGx is set to status active.<br>For detail, see **Section 16.7.1.2, TG status**. |
| 7 to 0 | PMMAnPFx | Indicates if TGm is set to status pending<br>0: TGx is not in pending status<br>1: TGx is set to status pending.<br>For detail, see **Section 16.7.1.2, TG status**. |

## 16.3.23 PMMAnSTR1 — PMM TG status register 1

This register indicates the transmit/receive status of TG.

**Access:** This register can be read in 32-bit units.

**Address:** <CSIHn_base> + $206C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMAnRF7 | PMMAnRF6 | PMMAnRF5 | PMMAnRF4 | PMMAnRF3 | PMMAnRF2 | PMMAnRF1 | PMMAnRF0 | PMMAnTF7 | PMMAnTF6 | PMMAnTF5 | PMMAnTF4 | PMMAnTF3 | PMMAnTF2 | PMMAnTF1 | PMMAnTF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 16.37 PMMAnSTR1 register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 15 to 8 | PMMAnRFx | Indicates the status of the reception path of TGx<br>0: Not all data are received for TGx.<br>1: PMM receive handling has finished for TGx.<br>**NOTES**<br>1. PMMAnRFx is set to 1 after the completion of TG reception.<br>2. When 1 is written to PMMAnTCTLx.PMMAnSPx while PMMAnTCNTx ≠ 0, PMMAnRFx remains set to 0 after the completion of last job data reception.<br>3. This bit does not indicate the status of the received data in RAM. If this bit is set it is only ensured that the last receive data was read by the DTS.<br>4. This bit is cleared when PMMAnSTC.PMMAnCLRFx is set to 1. In Disable mode, be sure to clear this bit by PMMAnSTC.PMMAnCLRFx. |
| 7 to 0 | PMMAnTFx | Indicates the status of the transmit path of TGx<br>0: Not all data are transferred for TGx.<br>1: PMM transmit handling has finished for TGx.<br>**NOTES**<br>1. PMMAnTFx is set to 1 after the completion of TG transmission.<br>2. When 1 is written to PMMAnTCTLx.PMMAnSPx while PMMAnTCNTx ≠ 0, PMMAnTFx remains set to 0 after the completion of last job data transmission.<br>3. This bit does not indicate the status of the communication inside PMMA. If this bit is set it is only ensured that all pending data are transferred to PMMA.<br>4. This bit is cleared when PMMAnSTC.PMMAnCLTFx is set to 1. In Disable mode, be sure to clear this bit by PMMAnSTC.PMMAnCLTFx. |

## 16.3.24 PMMAnSTC — PMM TG status clear register

This register clears the transmit/receive status of TG.

**Access:** This register can be write in 32-bit units.

**Address:** <CSIHn_base> + $2070_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by reset operation.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMAn CLRF7 | PMMAn CLRF6 | PMMAn CLRF5 | PMMAn CLRF4 | PMMAn CLRF3 | PMMAn CLRF2 | PMMAn CLRF1 | PMMAn CLRF0 | PMMAn CLTF7 | PMMAn CLTF6 | PMMAn CLTF5 | PMMAn CLTF4 | PMMAn CLTF3 | PMMAn CLTF2 | PMMAn CLTF1 | PMMAn CLTF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 16.38 PMMAnSTC register contents**

| Bit position | Bit name | Function |
|---|---|---|
| 31 to 16 | Reserved | When writing, write the value after reset. |
| 15 to 8 | PMMAnCLRFx | Clear the reception path status flag of TGx<br>0: no function.<br>1: Reset reception path status flag (PMMAnRFx = 0). |
| 7 to 0 | PMMAnCLTFx | Clear the transmit path status flag of TGx<br>0: no function.<br>1: Reset transmit path status flag (PMMAnTFx = 0). |

## 16.3.25 List of Caution

Table 16.39 Notes on Setting Registers (1/3)

| Register | Bit | Content |
|---|---|---|
| CSIHnCTL0 | CSIHnPWR | If this bit is cleared during communication, ongoing communication is aborted. A restart of the communication is then required. After cancelling the suspension, it is necessary to restart the communication.<br>Just after setting the PWR bit to 1 from 0, it is necessary to read CSIHnCTL0 register. |
| CSIHnCTL0 | CSIHnTXE<br>CSIHnRXE | Do not modify any of these bits while CSIHnCTL0.CSIHnPWR = 0. (These bits can be modified at the same time with the CSIHnCTL0.CSIHnPWR bit.)<br>Do not modify these bits while CSIHnSTR0.CSIHnTSF = 1, because ongoing communication is aborted and operation of this setting is not assured. |
| CSIHnCTL0 | CSIHnJOBE | Do not modify this bit while CSIHnCTL0.CSIHnPWR = 0.<br>This bit is only valid when CSIHnCTL1.CSIHnJE = 1.<br>Setting this bit is prohibited in slave mode. |
| CSIHnCTL0 | CSIHnMBS | This bit should be modified at the same time as the CSIHnCTL0.CSIHnPWR bit is set to 1.<br>Modification of this bit is only allowed while CSIHnSTR0.CSIHnTSF=0.<br>Do not modify of “FIFO mode <-> Direct Access mode” while CSIHnCTL0.CSIHnPWR=1. |
| CSIHnCTL1 | CSIHnSLRS | When the communication speed is faster than 5Mbps (excluding 5Mbps), CSIHnSLRS bit should be set to 1. |
| CSIHnCTL1 | CSIHnISCE | The CSIHnCFGx.CSIHnIDLx bit must be set to 0 if idle state control mode is enabled.<br>This bit can only be written if CSIHnCTL0.CSIHnPWR = 0 |
| CSIHnCTL1 | CSIHnCKR | Modification of this bit is only allowed while CSIHnCTL0.CSIHnPWR = 0.<br>This bit is only valid when the chip-select signal is not in use.<br>Fix it to 0 when the chip-select signal is in use.<br>Use this bit instead of CSIHnCFG0-7.CSIHnCKPn and CSIHnCFG0-7.CSIHnCKP must be set to “0”.<br>This bit must be set correctly in slave mode. |
| CSIHnCTL1 | CSIHnSLIT<br>CSIHnCSL[7:0]<br>CSIHnEDLE<br>CSIHnDCS<br>CSIHnCSRI<br>CSIHnHSE | Modification of these bits is only permitted while CSIHnCTL0.CSIHnPWR = 0. |
| CSIHnCTL1 | CSIHnSME<br>CSIHnJE<br>CSIHnLBM | Modification of these bits is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>Setting of this bit is prohibited in slave mode. |
| CSIHnCTL1 | CSIHnSSE | Modification of this bit is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>Setting of this bit is prohibited in master mode. |
| CSIHnCTL1 | CSIHnSIT | Modification of this bit is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>This bit is only valid in master mode. In slave mode, no delay is generated. |
| CSIHnCTL2 | CSIHnPRS[2:0] | Modification of this bit is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>Setting of the max baud rate is as follows.<br>• Master mode: CLKP_C/4<br>• Slave mode: CLKP_C/6 |
| CSIHnSTR0 | CSIHnSRP[7:0]<br>CSIHnSPF[7:0]<br>CSIHnFLF<br>CSIHnEMF<br>CSIHnTSF | Writing these bits is prohibited, and only reading is permitted. |
| CSIHnSTR0 | CSIHnTMOE<br>CSIHnOFE<br>CSIHnDCE<br>CSIHnPE<br>CSIHnOVE | Writing is invalid when CSIHnCTL0.CSIHnPWR=1.<br>Only read is effective.<br>Writing is effective when CSIHnCTL0.CSIHnPWR=0.These bits are initialized when CSIHnCTL0.CSIHnPWR=0-> 1 or CSIHnCTL0.CSIHnPWR=1-> 0. |
| CSIHnSTCR0 | CSIHnPCT | If this bit is set to1 during communication, ongoing communication is aborted. |

**Table 16.39 Notes on Setting Registers (2/3)**

| Register | Bit | Content |
|---|---|---|
| CSIHnMCTL0 | CSIHnMMS[1:0] | Modification of these bits is only allowed while CSIHnCTL0.CSIHnPWR = 0 and CSIHnCTL0.CSIHnMBS = 0. |
| CSIHnMCTL0 | CSIHnTO[4:0] | Modification of these bits is only allowed while CSIHnCTL0.CSIHnPWR = 0.<br>Set these bits to 0 in master mode.<br>Set these bits to 0 in direct access, dual buffer, and transmit-only buffer mode. |
| CSIHnMCTL1 | CSIHnFES[6:0]<br>CSIHnFFS[6:0] | Write to these bits during communication is permitted. |
| CSIHnMCTL2 | CSIHnBTST<br>CSIHnND[7:0]<br>CSIHnSOP[6:0] | Writing these bits is prohibited when CSIHnCTL0.CSIHnPWR = 0.<br>Writing these bits is prohibited when CSIHnCTL0.CSIHnTXE = CSIHnCTL0.CSIHnRXE = 0.<br>Writing these bits is prohibited when CSIHnSTR0.CSIHnTSF = 1.<br>Writing these bits is prohibited in direct access or FIFO mode. |
| CSIHnMRWP0 | CSIHnRRA[6:0] | Write to these bits during communication is permitted.<br>Writing these bits is prohibited in direct access or FIFO mode.<br>When writing is required, set “$0000_H$” to these bits in transmit only buffer mode. |
| CSIHnMRWP0 | CSIHnTRWA [6:0] | Write to these bits during communication is permitted.<br>Writing these bits is prohibited in direct access or FIFO mode. |
| CSIHnCFGx<br>x = 0 to 7 | CSIHnBRSSx [1:0]<br>CSIHnRCBx<br>CSIHnIDLx<br>CSIHnIDx[2:0]<br>CSIHnHDx[3:0]<br>CSIHnINx[3:0]<br>CSIHnSPx[3:0] | Modification of these bits is only allowed while CSIHnCTL0.CSIHnPWR = 0.<br>These bits must be set to 0 in slave mode. |
| CSIHnCFGx<br>x = 0 to 7 | CSIHnPSx[1:0]<br>CSIHnDLSx[3:0]<br>CSIHnDIRx<br>CSIHnDAPx | Modification of these bits is only allowed while CSIHnCTL0.CSIHnPWR = 0.<br>In slave mode, the CSIHnCFG0 setting is used for the configuration.<br>Therefore, all the bits in CSIHnCFG1 to CSIHnCFG7 must be set to 0. |
| CSIHnCFGx<br>x = 0 to 7 | CSIHnCKPx | Modification of this bit is only allowed while CSIHnCTL0.CSIHnPWR = 0.<br>As CSIHnCTL1.CSIHnCKR must be used, set this bit to 0 in slave mode.<br>If CS is not used, use CSIHnCTL1.CSIHnCKR bit instead of this bit, and clear this bit to 0. |
| CSIHnTX0W | CSIHnEOJ<br>CSIHnCIRE | Modification of the value of these bits are only permitted while CSIHnCTL0.CSIHnPWR=0.<br>This bit is only valid when CSIHnCTL1.CSIHnJE = 1.<br>While CSIHnCTL1.CSIHnJE = 0, the value of this bit is ignored even if 1 is read.<br>Set these bits to 0 in slave mode. |
| CSIHnTX0W | CSIHnEDL | This bit is only valid when CSIHnCTL1.CSIHnEDLE = 1.<br>While CSIHnCTL1.CSIHnEDLE = 0, the value of this bit is ignored even if 1 is read. |
| CSIHnTX0W | CSIHnSFNx | This bit is available in only CSIHnCTL1.CSIHnSME=1.<br>The IDLE period is set more than a period of CSIHnCFGx.CSIHnIDLx. |
| CSIHnTX0W | CSIHnISC | This bit is available in only CSIHnCTL1.CSIHnISCME=1.<br>Inactive period of CSIHTCSSn can set flexibly.<br>If using the idle state control function (CSIHnCTL1.CSIHnISCME), CSIHnCFGx.CSIHnIDLx must be 0. |
| CSIHnTX0W | CSIHnCSx | In master mode, setting these bits to “$FF_H$” is prohibited.<br>In slave mode, setting these bits to “$FE_H$” is needed. |
| CSIHnTX0W<br>CSIHnTX0H | | Reading to these bits during communication is prohibited in FIFO mode.<br>While CSIHnCTL0.CSIHnPWR = 0, read/write access these bits is prohibited in FIFO mode.<br>While CSIHnCTL0.CSIHnTXE = CSIHnCTL0.CSIHnRXE = 0, writing to these bits are prohibited in direct access mode. |

**Table 16.39 Notes on Setting Registers (3/3)**

| Register | Bit | Content |
|---|---|---|
| CSIHnRX0W<br>CSIHnRX0H | | Read access is permitted while CSIHnCTL0.CSIHnPWR=1.<br>While CSIHnCTL0.CSIHnPWR = 0, read access to these bits is prohibited in FIFO mode.<br>These bits are initialized when CSIHnCTL0.CSIHnPWR = 0 ->1 or CSIHnCTL0.CSIHnPWR = 1-> 0. |
| CSIHnBRSy<br>y = 0 to 3 | | Modification of this bit is only permitted while CSIHnCTL0.CSIHnPWR = 0.<br>Setting of the max baud rate is as follows with the CSIHnCTL2.CSIHnPRS[2:0] setting.<br>• Master mode: CLKP_C/4 (Up to 20 MHz bit/s)<br>• Slave mode: CLKP_C/6 (Up to 20 MHz bit/s) |

## 16.4 Interrupt Sources

CSIH can generate the following interrupt requests:

- CSIHTIC (communication interrupt)
- CSIHTIR (communication interrupt)
- CSIHTIRE (error interrupt)
- CSIHTIJC (job completion interrupt)

### 16.4.1 Overview

The error interrupt CSIHTIRE is generated when an error is detected. The generation of the other interrupts depends on the memory mode, the job mode, and – in case of the job completion interrupt CSIHTIJC –also the operating mode.

The job completion interrupt CSIHTIJC is only generated when job mode is enabled (CSIHnCTL1.CSIHnJE = 1). It is not available in slave mode.

The following table gives an overview.

**Table 16.40 Interrupt generation**

| | | Cause of interrupt | |
|---|---|---|---|
| **Memory mode** | **Interrupt** | **Job mode disabled CSIHnCTL1.CSIHnJE = 0** | **Job mode enabled CSIHnCTL1.CSIHnJE = 1** |
| FIFO | INTCSIHnTIC | Tx data empty*1 | Tx data empty*1 |
| | INTCSIHnTIR | Rx data full*2*5 | Rx data full*2*5 |
| | INTCSIHnTIRE | Error detected | Error detected |
| | INTCSIHnTIJC*3 | Not applicable | CSIHnTX0W.CSIHnCIRE = 1, or job abortion*4 |
| Transmit-only buffer, dual buffer | INTCSIHnTIC | End of communication | CSIHnTX0W.CSIHnCIRE = 1 and (CSIHnCTL0.CSIHnJOBE = 0 or CSIHnTX0W.CSIHnEOJ = 0) |
| | INTCSIHnTIR | Data received*5 | Data received*5 |
| | INTCSIHnTIRE | Error detected | Error detected |
| | INTCSIHnTIJC*3 | Not applicable | Job abortion*2 |
| Direct access | INTCSIHnTIC | One data transferred | One data transferred, if not aborted by job abortion*4 |
| | INTCSIHnTIR | Data received*5 | Data received*5 |
| | INTCSIHnTIRE | Error detected | Error detected |
| | INTCSIHnTIJC*3 | Not applicable | Job abortion*4 |

Note 1. “Tx data empty” refers to the FIFO fill level, defined by CSIHnMCTL1.CSIHnFES[6:0].
Note 2. “Rx data full” refers to the FIFO fill level, defined by CSIHnMCTL1.CSIHnFFS[6:0].
Note 3. INTCSIHnTIJC is not available in slave mode.
Note 4. Job abortion condition: CSIHnTX0W.CSIHnEOJ = 1 *and* CSIHnCTL0.CSIHnJOBE = 1
Note 5. Only if data reception is enabled (CSIHnCTL0.CSIHnRXE = 1).

### 16.4.2 General interrupt delay

In master mode, all interrupts generated by the master can be delayed by one half period of the transmission clock, CSIHTSCK. This is not possible in slave mode.

The delay is specified by setting bit CSIHnCTL1.CSIHnSIT = 1.

The following example illustrates the interrupt delay function, assuming a setting of CSIHnCTL1.CSIHnSIT = 1 (interrupt delay enabled), CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0 (normal clock and data phase), and CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$ (data length 8 bits).

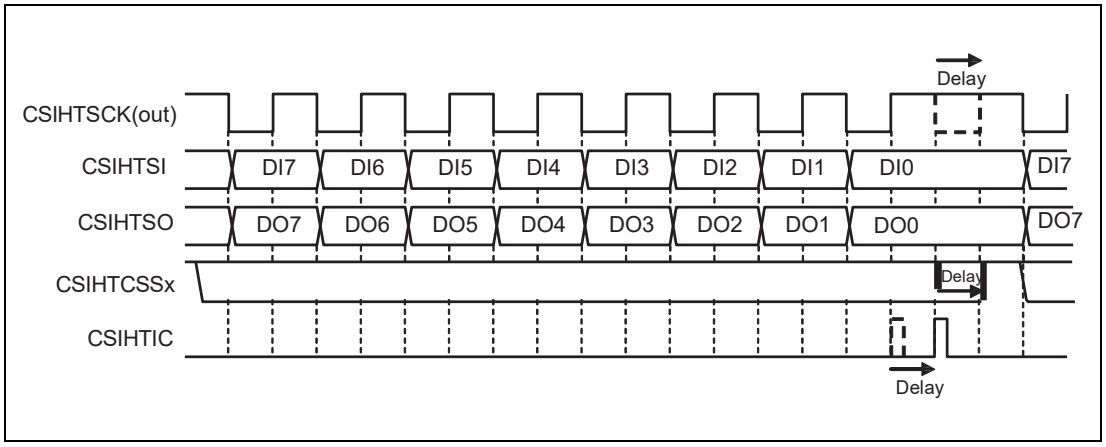



**Figure 16.2 Interrupt delay function (CSIHnCTL1.CSIHnSIT = 1)**

Setting CSIHnCTL1.CSIHnSIT = 1 adds half period delay to the transmission clock. This delays also the end of the present chip select signal (CSIHTCSSx).

### 16.4.3 CSIHTIC (communication interrupt)

Depending on the memory mode and the job mode, this interrupt is generated according to the conditions shown in the following table.

**Table 16.41 CSIHTIC interrupt generation**

| | Cause of interrupt | |
|---|---|---|
| **Memory mode** | **Job mode disabled CSIHnCTL1.CSIHnJE = 0** | **Job mode enabled CSIHnCTL1.CSIHnJE = 1** |
| FIFO | This interrupt is generated when transmission data is about to be missing in the FIFO, indicating to the application that new data should be added. CSIHTIC is generated, if the number of data to be sent remaining in the FIFO (CSIHnSTR0.CSIHnSPF[7:0]) equals CSIHnMTCL1.CSIHnFES[6:0].CSIHnTX0W. CSIHnCIRE = 1 and job abortion*1 are sent, the CSIHTIJC interrupt is generated instead of CSIHTIC | In the same way as when JE = 0, this interrupt is generated when the number of transmission data CSIHnSTR0.CSIHnSPF[7:0] remaining in the FIFO equals the CSIHnMCTL1.CSIHnFES[6:0] value, but this interrupt is not generated when job is aborted. |
| Transmit-only buffer, dual buffer | Generated at the End of communication. (Specified by the CSIHnMTLC2.CSIHnND[7:0] bit) | Generated when data with CSIHnTX0W.CSIHnCIRE = 1 is sent. Note that if data with CSIHnTX0W.CSIHnCIRE = 1 and job abortion*1 are sent, the CSIHTIJC interrupt is generated instead of CSIHTIC. |
| Direct access | Generated after every data transfer. | Generated after every data transfer, except when the communication was aborted. |

Note 1. Job abortion condition: CSIHnTX0W.CSIHnEOJ = 1 and CSIHnCTL0.CSIHnJOBE = 1.

#### 16.4.3.1 CSIHTIC in direct access mode

The examples below show the CSIHTIC behavior in direct access mode.

The examples assume:

- Master mode
- Direct access mode
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Normal clock and data phase
  (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- Data length 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$)
- Normal CSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)

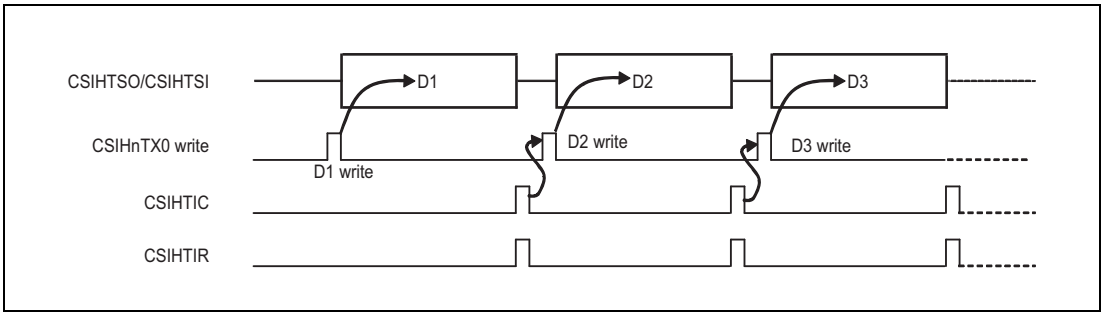



**Figure 16.3 Generation of CSIHTIC after transfer (CSIHnCTL1.CSIHnSLIT = 0)**

If job mode is enabled (CSIHnCTL1.CSIHnJE = 1) and a job ends because data is sent with CSIHnT0W.CSIHnEOJ = 1 and communication stop is requested (CSIHnCTL0.CSIHnJOBE = 1), then CSIHTIC is replaced by the job completion interrupt CSIHTIJC.

CSIHTIC can also be set up to occur as soon as the CSIHnTX0 register is free for the next data. This is specified by setting CSIHnCTL1.CSIHnSLIT = 1.

**NOTE**

This mode allows faster data transfer but is only available in direct access mode.

The effect is illustrated in the figure below.

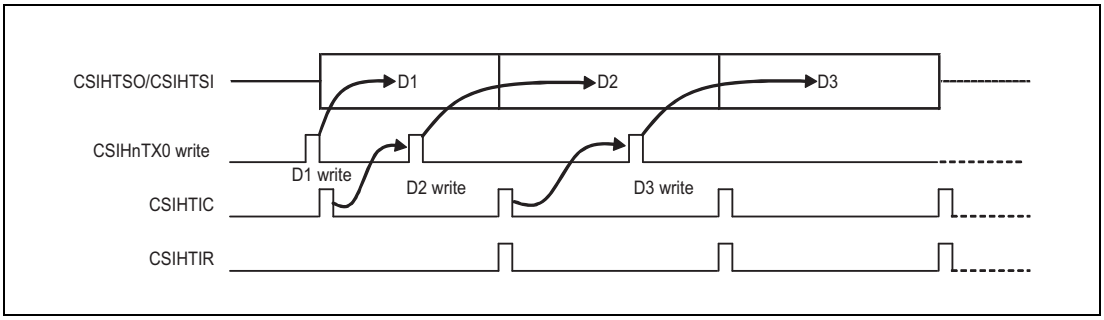



**Figure 16.4 Immediate generation of CSIHTIC (CSIHnCTL1.CSIHnSLIT = 1)**

Thus, the new data can be written in advance.

### 16.4.3.2 CSIHTIC in FIFO mode

The example below shows the CSIHTIC behavior in FIFO mode.

The example assumes:

- Master mode
- FIFO mode
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Normal clock and data phase
  (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- Data length 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$)

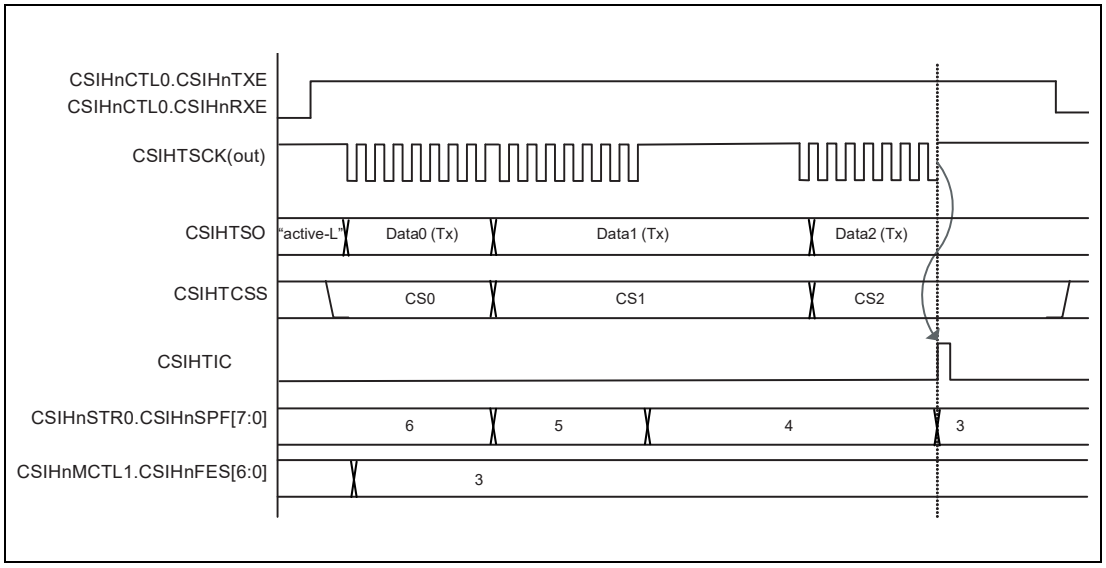



**Figure 16.5 Generation of CSIHTIC in FIFO memory mode**

The condition for “FIFO empty” is specified in CSIHnMCTL1.CSIHnFES[6:0]. In the example of the diagram above, the number of unsent data in FIFO is set to 3. CSIHnSTR0.CSIHnSPF[7:0] indicates the number of unsent data. When both match, the interrupt CSIHTIC occurs.

### 16.4.3.3 CSIHTIC in job mode

The example below shows the CSIHTIC behavior in job mode.

The example assumes:

- Master mode
- Job mode enabled (CSIHnCTL1.CSIHnJE = 1)
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Normal clock and data phase
  (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- Data length 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$)
- Normal CSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)

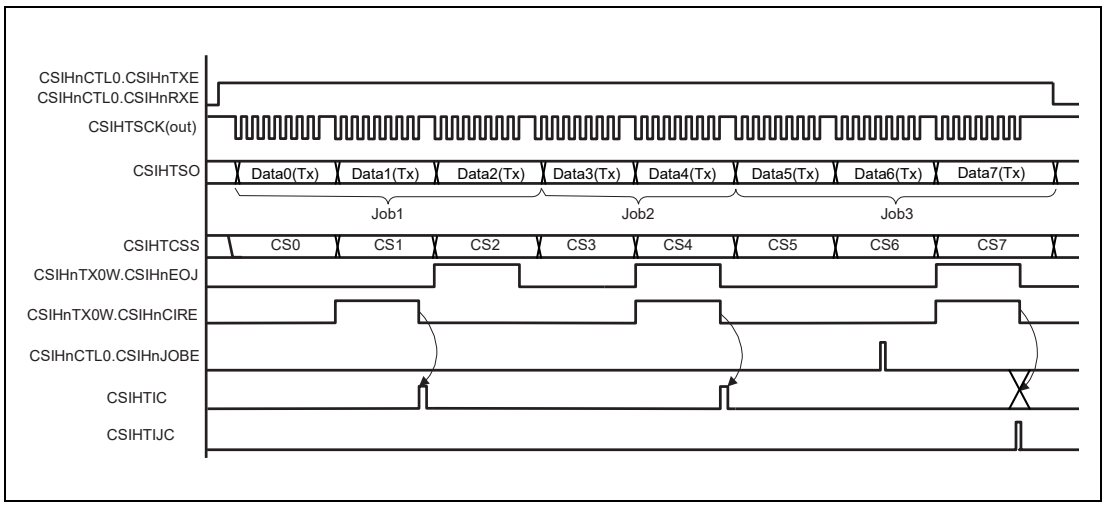



**Figure 16.6 Generation of CSIHTIC in job mode**

The rules for generating CSIHTIC in job mode are shown in the following table.

**Table 16.42 Generation of CSIHTIC in job mode**

| CSIHnTX0W. CSIHnEOJ | CSIHnTX0W. CSIHnCIRE | CSIHTIC |
|---|---|---|
| 0 | 0 | Not generated |
| 0 | 1 | Generated |
| 1 | 0 | Not generated |
| 1 | 1 | CSIHnCTL0.CSIHnJOBE = 0: Generated |
| | | CSIHnCTL0.CSIHnJOBE = 1: Not generated, replaced by interrupt CSIHTIJC |

#### 16.4.3.4 CSIHTIR (reception interrupt)

Depending on the memory mode and the job mode, this interrupt is generated according to the conditions below.

**Table 16.43 CSIHTIR interrupt generation**

| | Cause of interrupt | |
|---|---|---|
| **Memory mode** | **Job mode disabled CSIHnCTL1.CSIHnJE = 0** | **Job mode enabled CSIHnCTL1.CSIHnJE = 1** |
| FIFO | This interrupt occurs when the FIFO buffer is almost full with received data, indicating to the application that the FIFO must be emptied.<br>CSIHTIR is generated, if the number of received data in the FIFO (CSIHnSTR0.CSIHnSRP[7:0]) equals CSIHnMCTL1.CSIHnFFS[6:0]. | |
| Dual buffer | Generated when the communication has finished (as specified by the CSIHnMCTL2.ND[7:0] bit) | Generated after every data transfer. |
| Transmit-only buffer, Direct access | Generated after every data transfer. | |

### 16.4.3.5 CSIHTIR in direct access mode

The example below shows the CSIHTIR behavior in direct access mode.

The example below assumes:

- Master mode
- Direct access mode
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Normal clock and data phase
  (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- Data length 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$)

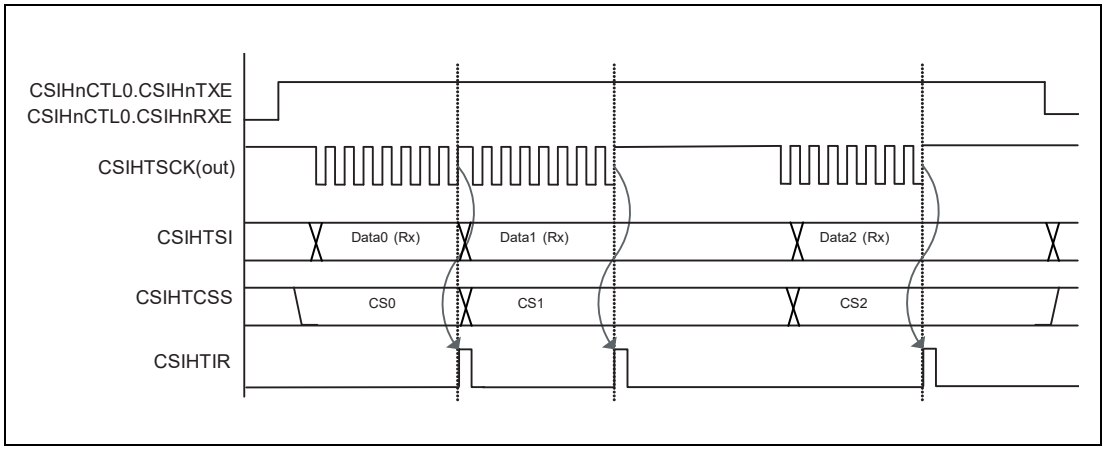



**Figure 16.7 Generation of CSIHTIR in direct access memory mode**

### 16.4.3.6 CSIHTIR in dual buffer mode

The example below shows the CSIHTIR behavior in dual buffer mode.

The example assumes:

- Master mode
- Dual buffer mode
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Default clock and data phase
  (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- Data length 8 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$)

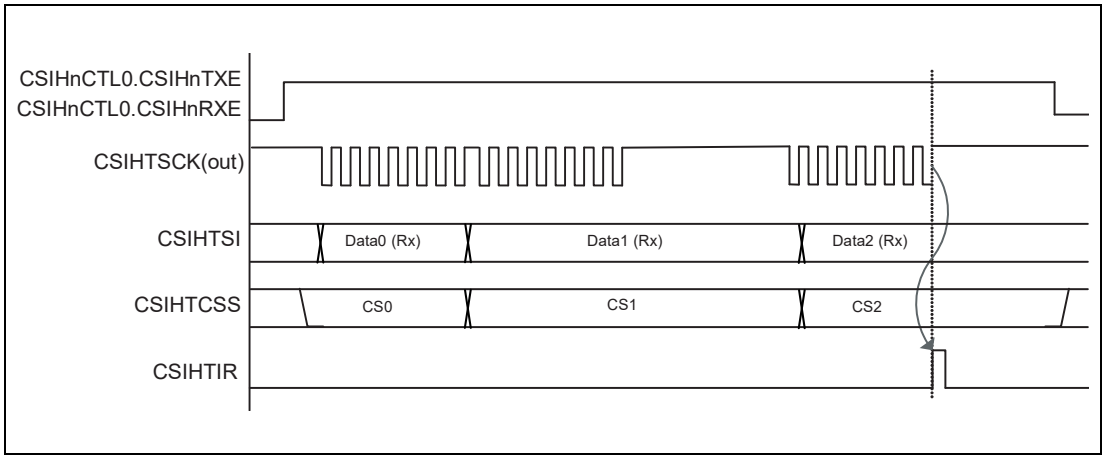



**Figure 16.8 Generation of CSIHTIR in dual buffer mode**

### 16.4.3.7 CSIHTIRE (reception error interrupt)

This interrupt is generated whenever an error is detected.

For details about generation interruption timing, see **Section 16.5.12, Error detection**.

**Table 16.44 Data error types**

| Error type | Communication status after error interrupt | Comment |
|---|---|---|
| FIFO overflow error | Interrupt is generated and communication continues | The data written to the FIFO is lost, but communication continues even if the error occurs |
| Parity error | Interrupt is generated and communication continues | — |
| Data consistency error | Interrupt is generated and communication continues | — |
| Time-out error | Interrupt is generated and communication continues | — |
| Overrun error | Interrupt is generated and communication continues<br><br>In slave mode interrupt is generated, but error can be avoided by hand-shake function. | This error is generated when the number of received data items is 0 and the CPU attempts to read reception data and data was received, but FIFO is full. (This is only valid in FIFO mode) In transmit-only buffer mode or direct access mode: Received data is left in the CSIHnRX0 register. |

The type of error that caused the generation of CSIHTIRE is flagged in register CSIHnSTR0.

Additionally a parity error flag and a data consistency error flag are attached to the received data in CSIHnRX0W.

For details about the various error types, see **Section 16.5.12, Error detection**.

### 16.4.3.8 CSIHTIJC (job completion interrupt)

This interrupt supports the handling of jobs, see **Section 16.5.3.3, Job concept**. This interrupt is only available in master mode.

Job mode is enabled by setting CSIHnCTL1.CSIHnJE = 1. When CSIHnCTL1.CSIHnJE = 0, CSIHTIJC is not generated.

Depending on the memory mode, this interrupt is generated according to the condition shown in the following table.

**Table 16.45 CSIHTIJC interrupt generation**

| Memory mode | Job mode disabled CSIHnCTL1.CSIHnJE = 0 | Job mode enabled CSIHnCTL1.CSIHnJE = 1 |
|---|---|---|
| FIFO | Not applicable | Indicates that the communication has stopped at the end of a job after a job abortion*1 was triggered |
| Transmit-only buffer | | |
| Dual buffer | | |
| Direct access | | |

Note 1. Job abortion condition: CSIHnTX0W.CSIHnEOJ = 1 and CSIHnCTL0.CSIHnJOBE = 1

# 16.5 Operation

## 16.5.1 Operating modes (master/slave)

### 16.5.1.1 Master mode

In master mode, the BRG frequency can be set by combining the CSIHnCTL2.CSIHnPRS[2:0] bit and the CSIHnBRSy.CSIHnBRSy[11:0] bit.

Master mode is enabled by setting CSIHnCTL2.CSIHnPRS[2:0] to anything but $111_B$. In master mode, the BRG frequency can be set by combining the CSIHnCTL2.CSIHnPRS[2:0] bit and the CSIHnBRSy.CSIHnBRSy[11:0] bit.

#### (1) Chip select signals

In master mode, one or several chip select signals can be used. If several slaves are connected to the master, the chip select signals can be used to address one or several of the slaves. Only a selected slave is then enabled for communication.

The communication protocol as well as additional parameters is stored separately for each chip select signal. This makes it possible to adapt the data transfer individually to the requirements of each slave. For details see **Section 16.5.3, Chip selection (CS) features**.

#### (2) Clock defaults

The default level of CSIHTSCK depends on the clock phase selection bit: It is high when CSIHnCTL1.CSIHnCKR = 0, and is low when CSIHnCTL1.CSIHnCKR = 1.

The example below shows the communication in master mode for 8 data bits, CSIHnCTL1.CSIHnCKR = 0, CSIHnCFGx.CSHBAnDAPx = 0, and MSB first:

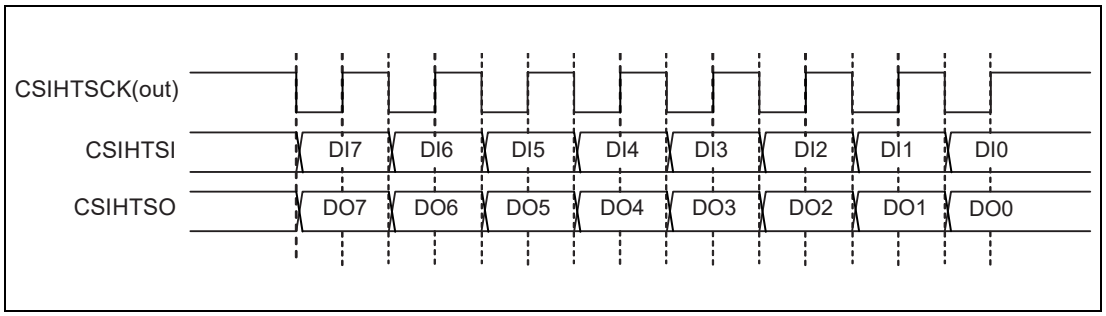



**Figure 16.9 Transmit/receive in master mode**

### 16.5.1.2 Slave mode

In slave mode, another device is the communication master and provides the transmission clock. Send/receive operation normally starts as soon as a clock signal is detected.

Slave mode is selected by setting the CSIHnCTL2.CSIHnPRS[2:0] bit to $111_B$.

In slave mode, the transmission protocol setting of the CSIHnCFG0 register is enabled (setting of the CSIHnCFG1-CSIHnCFG7 register is disabled).

- CSIHnPS0[1:0]: parity usage
- CSIHnDLS0[3:0]: data length selection
- CSIHnDIR0: data direction
- CSIHnCKP0,CSIHnDAP0: clock and data phase

**NOTE**

When using slave mode, disable the baud rate generator (BRG) by clearing bit CSIHnBRSy.CSIHnBRSy[11:0].

But only when time-out error detection is used, set some value these bits as the master does. Because these bits are used as time-out clock.

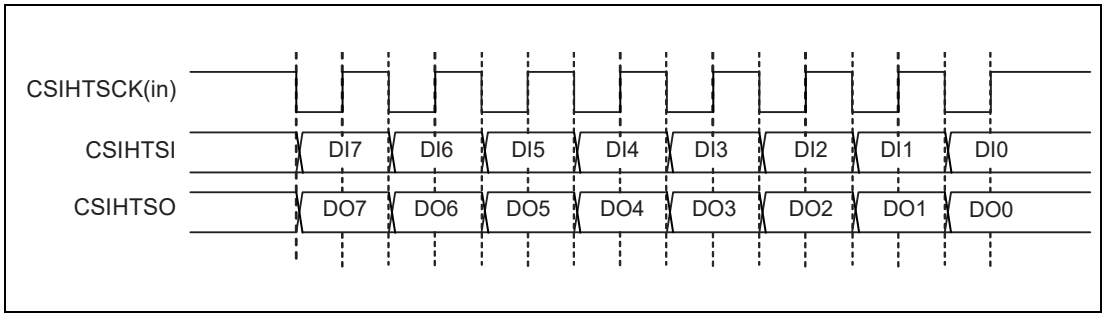



**Figure 16.10 Transmit/receive in slave mode**

## 16.5.2 Master/slave connections

### 16.5.2.1 One master and one slave

The following figure illustrates the connections between one master and one slave.

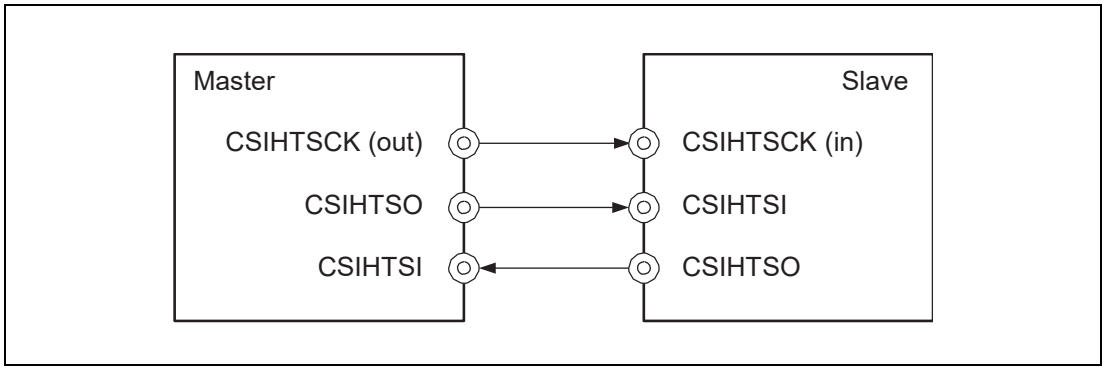



**Figure 16.11 Direct master/slave connection**

### 16.5.2.2 One master and multiple slaves

The following figure illustrates the connections between one master and multiple slaves. In this example, the master provides one chip select (CS) signal to each of the slaves. This signal is connected to the slave select input CSIHTSSIZ of the slave.
The recognition function of the CSIHTSSIZ signal can be enabled/disabled by bit CSIHnCTL1.CSIHnSSE.

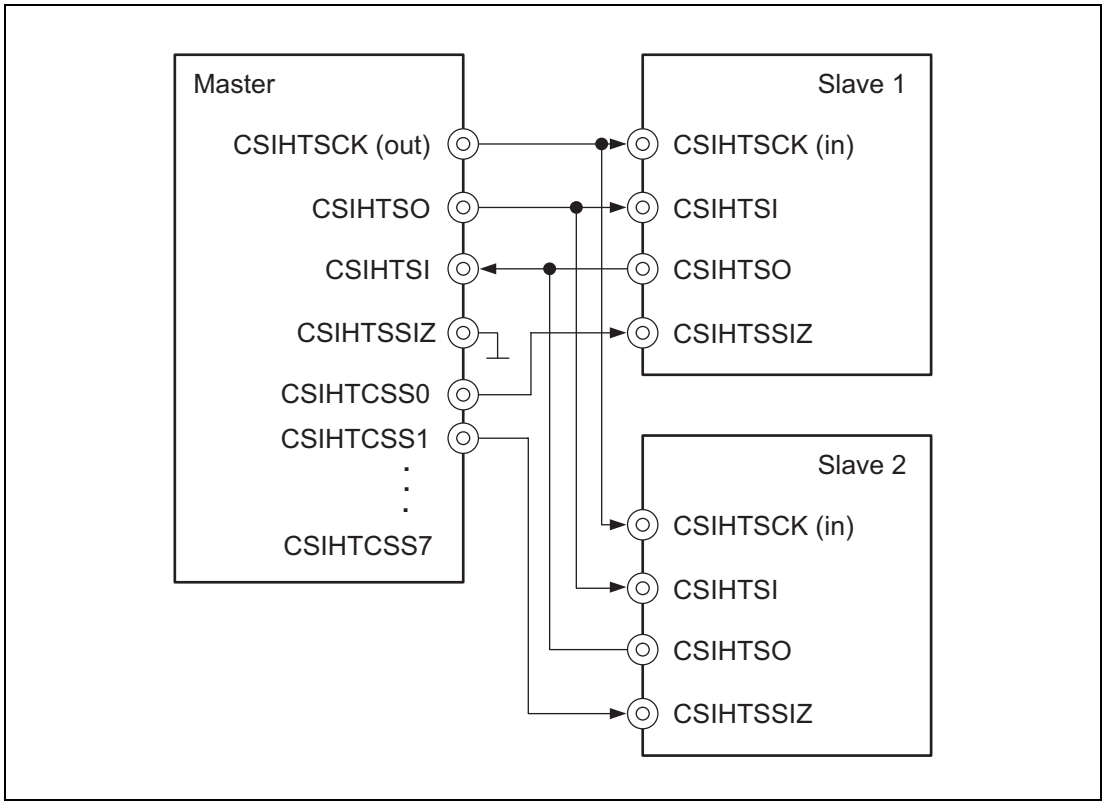



**Figure 16.12 Master to multiple slaves connection**

By default, the chip select level is active low. That means, a slave is selected (enabled) as a CSIH slave when its CSIHTSSIZ signal has low level. However, to adapt the CS to other devices, the output level of each chip select signal can also be programmed to be active high.

If a slave is not selected, it will neither receive nor transmit data. In addition, its output CSIHTSO of a slave that is not selected is set to input mode in order to avoid interference with the output of another slave that was selected.

**CAUTION**

**I/O function of chip SCKO pin must be controlled by not only CSIHTSCO but also control signal of each product. Because reset (system reset) value of CSIHTMD is active level. I/O function of chip SO pin must be controlled by not only CSIHTSCO but also control signal of each product. Because reset (system reset) value of CSIHTSCO is active level.**

### 16.5.3 Chip selection (CS) features

The chip select signal, CSIHTCSSx can be used by the master to select one or several slaves for communication.

#### 16.5.3.1 Configuration registers

The parameters for each chip select signal CSIHTCSSx are defined in the corresponding configuration register CSIHnCFGx. The parameters include the communication protocol and additional CS parameters.

The communication protocol specifies:

- Data length: The number of bits to be sent or received.
- Transfer direction: MSB or LSB first.
- Parity usage: Odd, even, 0 parity or none.
- Clock phase and data phase.

Additional parameters for each chip select signal that is only available in master mode are:

- One out of four baud rates can be selected for each chip select.
- Chip select priority: Separates between “dominant” and “recessive” chip select signals. The priority applies if two or more chip selects signals with different configurations are simultaneously activated for message broadcasting. In this case, the configuration that is set as dominant is used.

The principle is also called “Recessive Configuration for Broadcasting” (RCB).

**CAUTION**

**It is forbidden to specify several chip select signals as dominant with different configurations unless all dominant chip selects have the same configuration.**

- Chip select timing:
  - Setup time $T_{setup}$: The time from setting the CS signal active to starting data output.
  - Inter-data time $T_{inter}$: The time between data packets while the same CS signal is active.
  - Hold time $T_{hold}$: Hold time of CS active level before changing the CS.
  - Idle time $T_{idle}$: Inactive time after terminating a CS signal or after every data transfer to the same CSx.

The CS timings of the setup time, the inter-data time, the hold time, and the idle time are illustrated in the figure below. If the CSIHnTX0W.CSIHnCSx setting of two consecutive transfers is different, an idle state is inserted between two transfers.

**Figure 16.13** provides an example when the default CS1 and CS2 signals are active low (CSIHnCTL1.CSIHnCSL1 bit = 0, CSIHnCTL1.CSIHnCSL2 bit = 0). The active level can be specified individually for each CS.

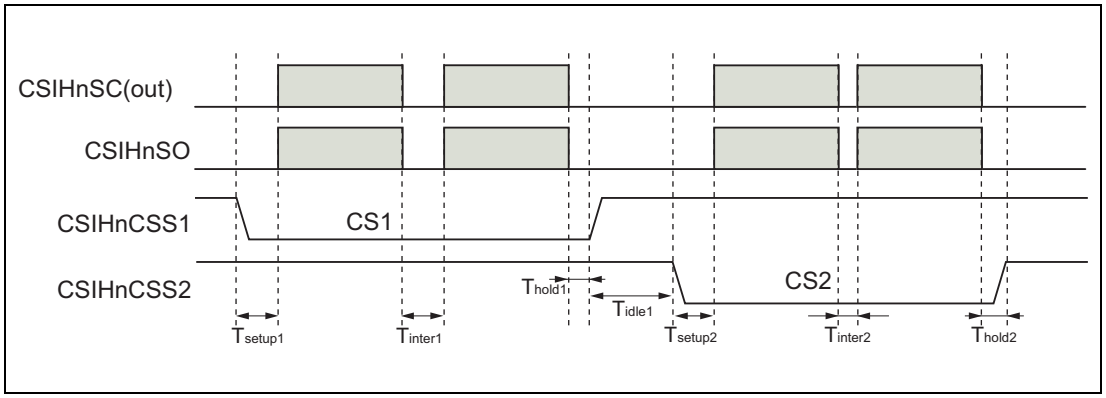



**Figure 16.13 Chip select timings**

### 16.5.3.2 CS example

The following figure shows an example of two consecutive data transmissions.

The first communication uses CS0 to communicate with one single slave. The second enables CS0 and CS1 to broadcast a message to two slaves. The priority of CS0 is set to “recessive: low priority”, the priority of CS1 to “dominant: high priority”.

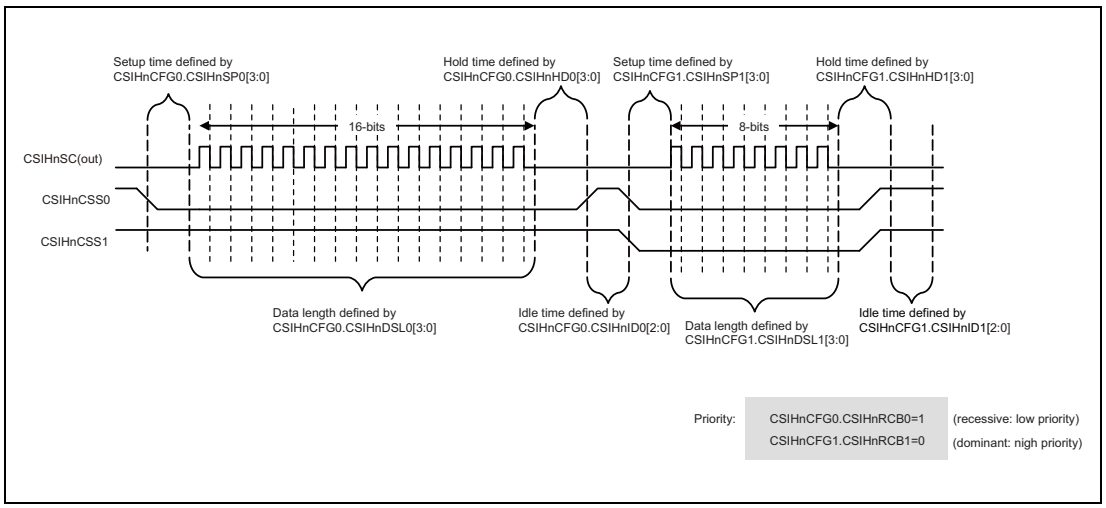



**Figure 16.14 Chip select and RCB example**

Note that each CS signal can have a different value for setup, inter-data time, hold time, and idle time.

A particular chip select signal is activated by writing ‘0’ to the appropriate bit in the transmission register CSIHnTX0W.CSIHnCSx.

CSIHnRX0W.CSIHnCSx in the reception register indicates the chip select signal associated with the received data.

During broadcasting, chip selects which are simultaneously active with the same RCB value must have the same configuration.

### 16.5.3.3 Job concept

In terms of CSIH, a job consists of a number of data pockets that are transferred.

#### Job mode enable

The job mode can only be enabled in master mode. The job mode is enabled and disabled by CSIHnCTL1.CSIHnJE.

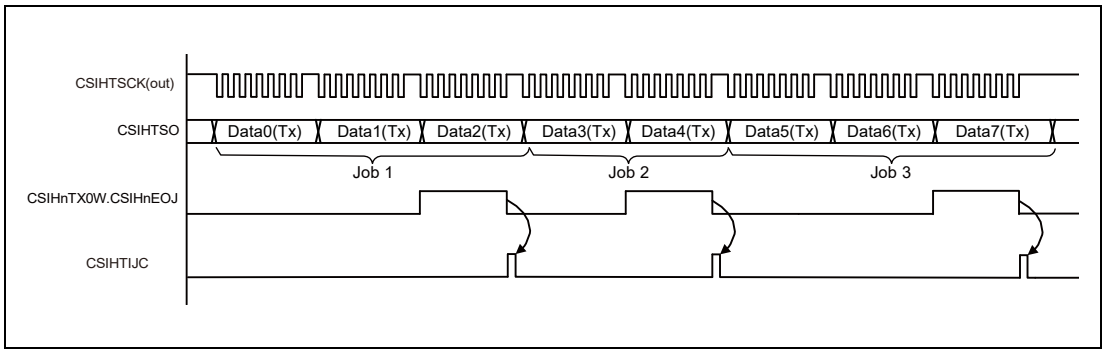



**Figure 16.15 Job examples**

A job ends when a data packet with the end-of-job bit set, i.e. by transmitting a data packet with CSIHnTX0W.CSIHnEOJ = 1.

A communication stop can be specified to occur after a job has finished. This is done by setting CSIHnCTL0.CSIHnJOBE. When CSIHnJOBE is set, the communication continues until data is sent, for which the CSIHnEOJ bit was set. After this data is sent, the communication is stopped and the - interrupt CSIHTIJC is generated.

## 16.5.4 Chip select timing details

### 16.5.4.1 Changing the clock phase

The serial clock level specified by CSIHnCFGx.CSIHnCKPx may be changed when communication is disabled. The minimum value of an idle time is one period of transmission clock (CSIHTSCO).

If the idle time is set to 0.5 transmission clock periods (in CSIHnCFGx.CSIHnIDx[2:0]) and two consecutive data packets are sent with different CSIHnCFGx.CSIHnCKPx configuration, the idle time is automatically extended to one period of CSIHTSCO.

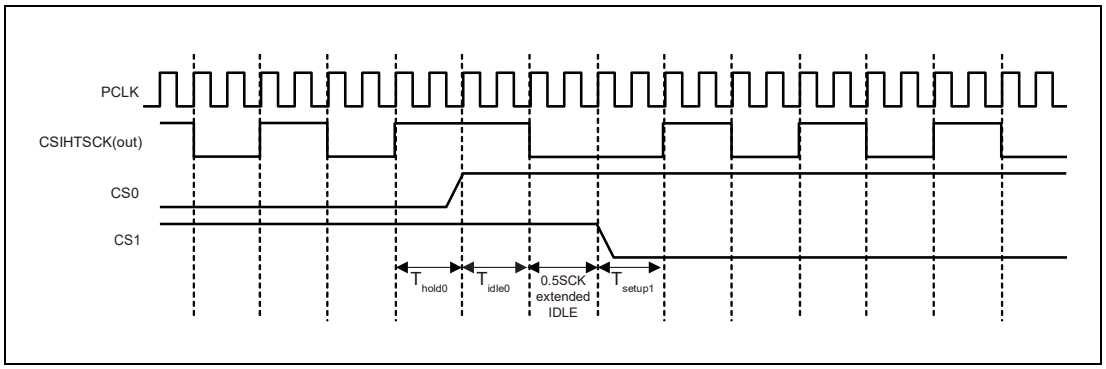



**Figure 16.16 Clock phase timing**
**CLKP_C/4, $T_{hold}0 = T_{setup1}$ = 0.5SCK, $T_{idle0}$ = 0.5SCK, CKP0 = 0 (CS0) -> CKP1 = 1 (CS1)**

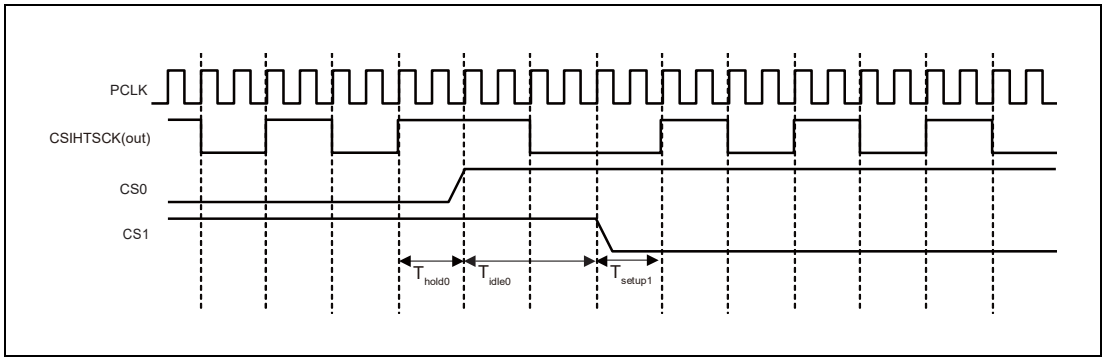



**Figure 16.17 Clock phase timing**
**CLKP_C/4, $T_{hold}0 = T_{setup1}$ = 0.5SCK, $T_{idle0}$ = 1SCK, CKP0 = 0 (CS0) -> CKP1 = 1 (CS1)**

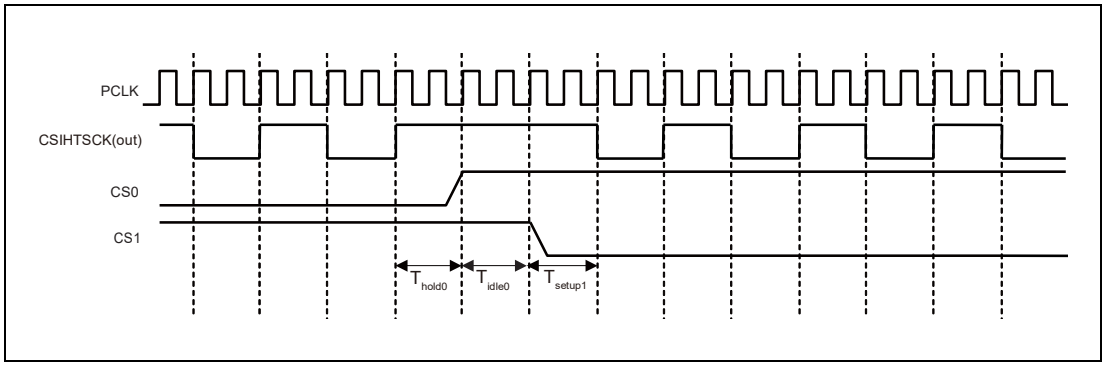



**Figure 16.18 Clock phase timing**
**CLKP_C/4, $T_{hold}0 = T_{setup1} = 0.5SCK$, $T_{idle0} = 0.5SCK$, CKP0 = 0 (CS0) -> CKP2 = 0 (CS2)**

### 16.5.4.2 Changing the data phase

The CSIHnCFGx.CSIHnDAPx bit defines the phase of the data bits relative to the clock.

SCK level changes whether the value of CSIHnCKPx bit for the next frame.

There is a difference of 0.5SCK until the edge of the serial clock is output by setting of CSIHnCFGx.CSIHnDAPx

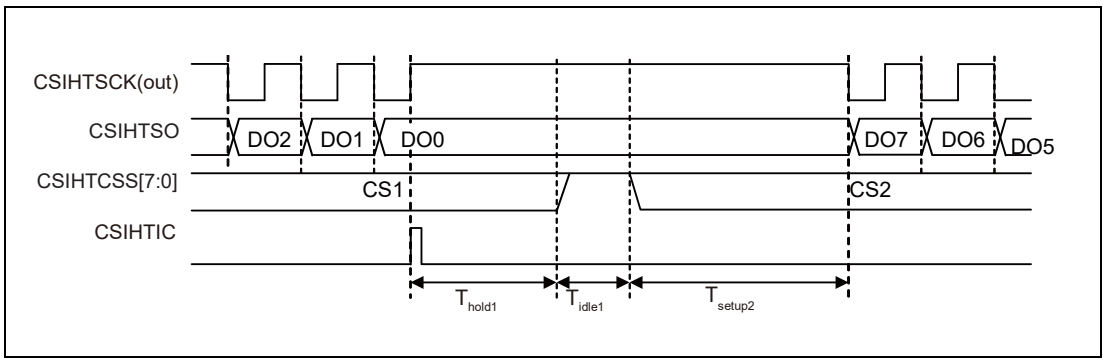



**Figure 16.19 Data phase timing with CSIHnCFG1.CSIHnCKP1 = 0, CSIHnCFG1.CSIHnDAP1 = 0 and CSIHnCFG2.CSIHnCKP2 = 0, CSIHnCFG2.CSIHnDAP2 = 0**

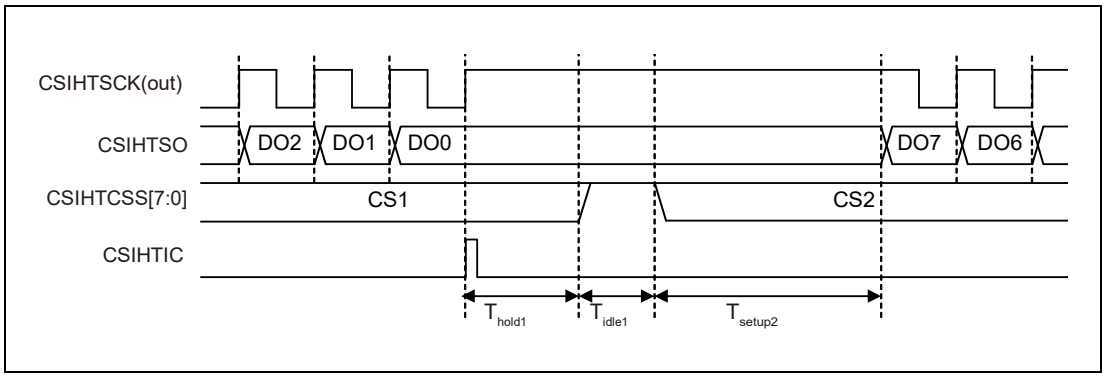



**Figure 16.20 Data phase timing with CSIHnCFG1.CSIHnCKP1 = 0, CSIHnCFG1.CSIHnDAP1 = 1 and CSIHnCFG2.CSIHnCKP2 = 0, CSIHnCFG2.CSIHnDAP2 = 1**

If the CSIHnCFGx.CSIHnDAPx bit changes between two consecutive chip select signals, the data phase of CSIHnSO changes during the idle period after the last bit of the first data is transferred:

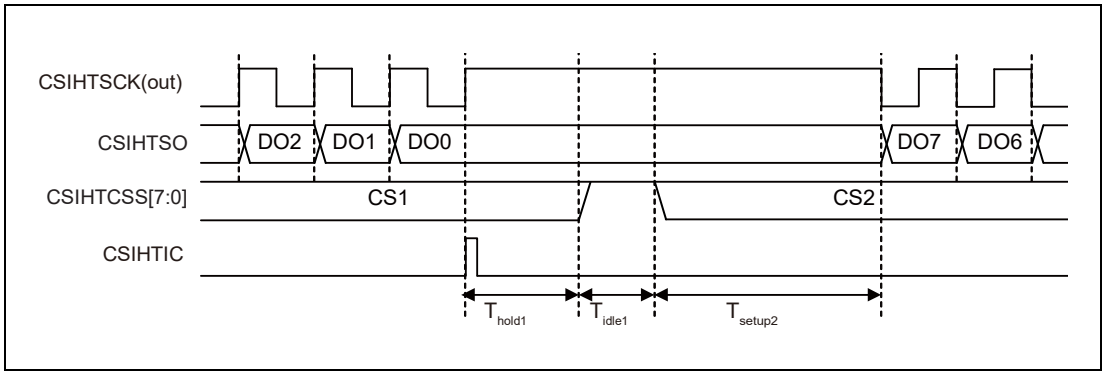



**Figure 16.21 Data phase timing with CSIHnCFG1.CSIHnCKP1 = 0, CSIHnCFG1.CSIHnDAP1 = 1 and CSIHnCFG2.CSIHnCKP2 = 0, CSIHnCFG2.CSIHnDAP2 = 0**

Note that the minimum idle time of one CSIHnSC period is automatically inserted, if CSIHnCFGx.CSIHnIDx[2:0] = 0 ($T_{idle1}$ = 0.5 transmission clock periods).

### 16.5.5 Transmission clock selection

In master mode, the transmission baudrate is selectable using the following bits:

- CSIHnCTL2.CSIHnPRS[2:0]
- CSIHnBRSy.CSIHnBRSy[11:0] (y = 0 to 3)
- CSIHnCFGx.CSIHnBRSSx[1:0] (x = 0 to 7)

While the settings in the CSIHnCTL2 register determine the transmission clock CSIHTCLK, a chip select dedicated prescaler, controlled by CSIHnCFGx.CSIHnPSCLx[1:0], allows to generate different baudrates for different chip selects.

The following figure shows a block diagram of the baudrate generator.

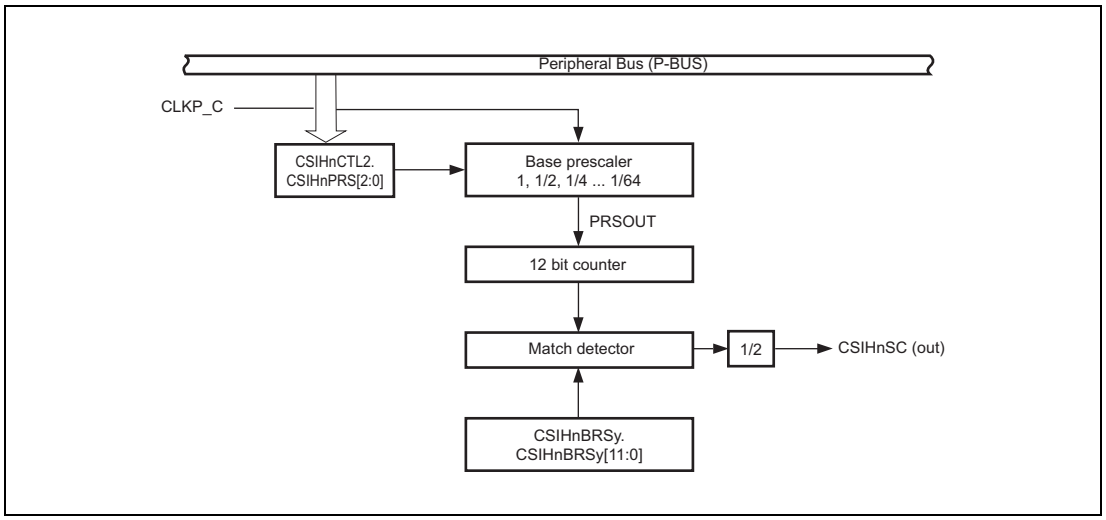



**Figure 16.22 Baud rate generator block diagram**

Clearing CSIHnBRSy.CSIHnBRSy[11:0] disables the baudrate generator, and thus all CSIHTSCK are stopped.

#### Baud rate calculation

The baudrate is calculated as:

$$CSIHTSCK = CLKP_C / (2^{\alpha} \times k \times 2)$$

where

$\alpha$ = CSIHnCTL2.CSIHnPRS[2:0] = 0 to 6

k = CSIHnBRS0.CSIHnBRS0[11:0] = 1 to 4095
(when CSIHnBRSSx[1:0] = 0)

k = CSIHnBRS1.CSIHnBRS1[11:0] = 1 to 4095
(when CSIHnBRSSx[1:0] = 1)

k = CSIHnBRS2.CSIHnBRS2[11:0] = 1 to 4095
(when CSIHnBRSSx[1:0] = 2)

k = CSIHnBRS3.CSIHnBRS3[11:0] = 1 to 4095
(when CSIHnBRSSx[1:0] = 3)

### Baud rate limits

When setting the baud rate, please note:

- Maximum acceptable baud rate in master mode is CLKP_C/4.
- Maximum acceptable baud rate in slave mode is CLKP_C/6 (must be ensured by the external master).
- Minimum baud rate in both modes is CLKP_C/524160.

CLKP_C for CSIH is 160 MHz and the maximum baud rate is as follows:

- 20.0 Mbps in master mode
- 20.0 Mbps in slave mode

The minimum baud rate is 305.25 bps (CLKP_C/524160).

### 16.5.6 CSIH buffer memory

The CSIH has a configurable RAM that can be used for buffered I/O. The size is 128 words. One word is comprised of 32 bits data plus 7 bits ECC.

The following configurations are available:

| Mode | CSIHnCTL0. CSIHnMBS | CSIHnMCTL0. CSIHnMMS[1:0] |
|---|---|---|
| FIFO mode | 0 | $00_B$ |
| Dual buffer mode | | $01_B$ |
| Transmit-only buffer mode | | $10_B$ |
| Direct access mode | 1 | X |

#### 16.5.6.1 FIFO mode

In FIFO mode, data can be written to the CSIHnTX0W register without waiting for completion of the transmission, and data can be received without reading the CSIHnRX0W register immediately, provided the FIFO is not full.

Data to be transmitted is stored to the FIFO memory. Transmission and reception occur simultaneously – one data is sent, one data is received. That means, received data overwrites the transmitted data in the FIFO.

The CSIH automatically updates the respective FIFO memory pointers when a data packet is committed, sent or received:

**Table 16.46 FIFO mode**

| Pointer description | Control bit[*1] | Range |
|---|---|---|
| Number of unsent words | CCSIHnSTR0.CSIHnSPF[7:0] | 0 to 128 |
| Number of words received and stored in the FIFO | CSIHnSTR0.CSIHnSRP[7:0] | 0 to 128 |
| Address of data to be sent | CSIHnMRWP0.CSIHnTRWA[6:0] | $0000_H$ to $01FC_H$ |
| Address of received data | CSIHnMRWP0.CSIHnRRA[6:0] | $0000_H$ to $01FC_H$ |
| Address of currently transferred/ received data | CSIHnMCTL2.CSIHnSOP[6:0] | $0000_H$ to $01FC_H$ |

Note 1. Pointers are automatically incremented after each read/write.

The CSIH status register contains also two FIFO status flags:

- CSIHnSTR0.CSIHnFLF: FIFO full
- CSIHnSTR0.CSIHnEMF: FIFO empty

When this mode is started, bit CSIHnSTCR0.CSIHnPCT must be set. This resets all FIFO pointers and flags.

#### 16.5.6.2 Dual buffer mode

In this mode, the memory is divided into two parts of equal size – this means 64 words for transmit data and 64 words for received data. In dual buffer mode, the respective buffer pointers indicate the values shown in the following table.

**Table 16.47 Dual buffer mode**

| Pointer description | Pointer*1 | Range |
|---|---|---|
| Destination address for data written to or read from CSIHnTX0W/H | CSIHnMRWP0.CSIHnTRWA[6:0] | 0 to 63 |
| Address of data read from CSIHnRX0W/H | CSIHnMRWP0.CSIHnRRA[6:0] | 0 to 63 |

Note 1. Pointers are automatically incremented after each read/write.

#### 16.5.6.3 Transmit-only buffer mode

In this mode, the entire memory is used to save transmission data.

Received data must be read directly from CSIHnRX0W/H.

In transmit-only buffer mode, the respective buffer pointer indicates the values shown in the following table.

**Table 16.48 Transmit-only buffer mode**

| Pointer description | Pointer*1 | Range |
|---|---|---|
| Destination address for data written to or read from CSIHnTX0W/H | CSIHnMRWP0.CSIHnTRWA[6:0] | 0 to 127 |

Note 1. Pointers are automatically incremented after each read/write.

#### 16.5.6.4 Direct access mode

In direct access mode, the CSIH memory is completely bypassed:

- Transmission data provided by the CPU to the transmission register CSIHnTX0W or CSIHnTX0H is directly copied to the shift register.
- Reception data is directly copied from the shift register to the reception register CSIHnRX0W or CSIHnRX0H.

While the high priority communication is executed by CPU control, operates as same as direct access mode regardless of the CSIHnMSB bit setting.

## 16.5.7 Data transfer modes

### 16.5.7.1 Transmit-only mode

Setting CSIHnCTL0.CSIHnTXE = 1 and CSIHnCTL0.CSIHnRXE = 0 puts the CSIH in transmit-only mode. Start of transmission depends on the memory mode:

- In case of FIFO or direct access mode, transmission starts when transmit data is written to the CSIHnTX0W or CSIHnTX0H register.
- In case of dual buffer or transmit-only buffer mode, transmission starts when bit CSIHnMCTL2.CSIHnBTST is set.

### 16.5.7.2 Receive-only mode

Setting CSIHnCTL0.CSIHnTXE = 0 and CSIHnCTL0.CSIHnRXE = 1 puts the CSIH in receive-only mode.

In master mode, start of reception depends on the memory mode:

- In case of FIFO or direct access mode, reception starts when dummy data is written in the CSIHnTX0W or CSIHnTX0H register.
- In case of dual buffer mode or transmit-only buffer mode, reception starts when bit CSIHnMCTL2.CSIHnBTST is set.

In slave mode, reception starts as soon as the CSIHTSCK transmission clock from the master is received. It is not necessary to write data to the CSIHnTX0W or CSIHnTX0H register of the slave.

### 16.5.7.3 Transmit/receive mode

Setting CSIHnCTL0.CSIHnTXE = 1 and CSIHnCTL0.CSIHnRXE = 1 puts the CSIH in transmit/receive mode.

Start of communication (transmission and reception) depends on the memory mode:

- In case of FIFO or direct access mode, communication starts when transmit data is written to the CSIHnTX0W or CSIHnTX0H register.
- In case of dual buffer or transmit-only buffer mode, communication starts when bit CSIHnMCTL2.CSIHnBTST is set.

### 16.5.7.4 Summary

The following table summarizes this section. It shows how data transfer is started in the various memory, operating, and transfer modes.

**Table 16.49 Start of data transfer**

| Memory and operating mode | | Transfer mode | |
|---|---|---|---|
| | | Transmit-only<br>Transmit / receive | Receive-only |
| FIFO, direct access | Master | Writing to the CSIHnTX0W register or the CSIHnTX0H register | Writing to the CSIHnTX0W register on the CSIHnTX0H register |
| | Slave | | Incoming clock from the master |
| Transmit-only buffer, dual buffer | Master | CSIHnMCTL2.CSIHnBTST = 1 | CSIHnMCTL2.CSIHnBTST = 1 |
| | Slave | | Incoming clock from the master |

## 16.5.8 Data length selection

### 16.5.8.1 Data length between 2 and 16 bits

The length of a data packet is selectable for each chip select signal from 2 to 16 bits using CSIHnCFGx.CSIHnDLSx[3:0]. The examples below show the communication with MSB first (CSIHnCFGx.CSIHnDIRx = 0).

Data length = 16 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $0000_B$)

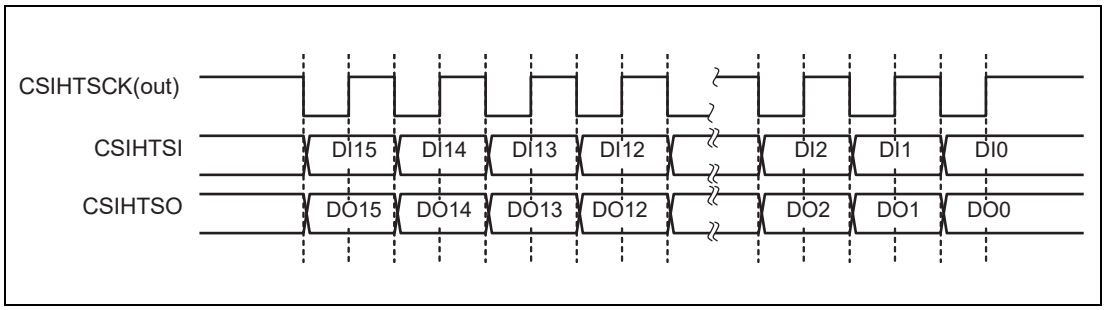



**Figure 16.23 16 bit data length, MSB first**

Data length = 14 bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1110_B$):

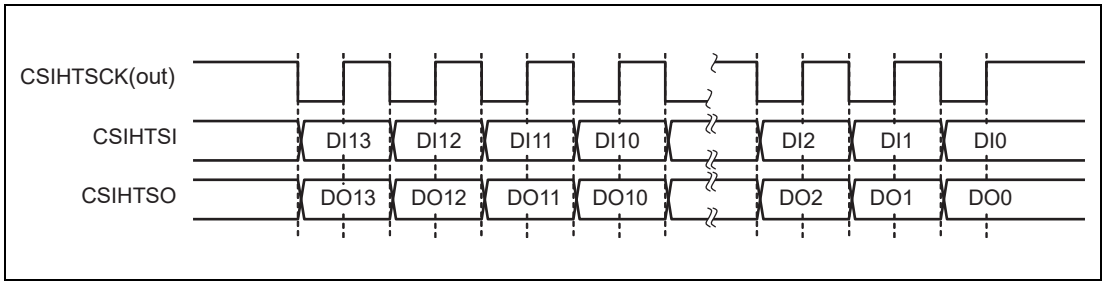



**Figure 16.24 14 bit data length, MSB first**

**CAUTION**

**When the data length is set to 2, the handshake function might be restricted. For details, see Section 16.5.11, Handshake function.**

### 16.5.8.2 Data length greater than 16 bits

If the length of the data to be sent/received exceeds 16 bits, the extended data length (EDL) feature can be used.

EDL is enabled by setting bit CSIHnCTL1.CSIHnEDLE to 1.

EDL works as follows:

- The data has to be broken into 16-bit blocks plus remainder. For example, a string of 42 bits would be broken into two 16-bit blocks plus 10 bits.
- The remainder defines the “data length” that has to be specified in CSIHnCFGx.CSIHnDLSx[3:0].
- For transmitting the 16-bit blocks, CSIHnTX0W.CSIHnEDL must be set to 1. In this case, the data written to CSIHnTX0W is sent as a 16-bit data length regardless of the CSIHnCFG0.CSIHnDLS[3:0] bit setting.
- The transfer is complete after a block with the specified data length (the remainder of data specified with CSIHnTX0W.CSIHnEDL = 0) has been sent.

#### Example

Example for sending 40-bit data ($123456789A_H$) to CS0:

40 bits are split 2 × 16 bits plus 8 bits.

- Initialize CSIHnCFG0.CSIHnDLS0[3:0] = 8.
- To send $123456789A_H$ with MSB first, write the following sequence to CSIHnTX0W:
  - $20FE\ 1234_H$ (CSIHnTX0W.CSIHnEDL = 1)
  - $20FE\ 5678_H$ (CSIHnTX0W.CSIHnEDL = 1)
  - $00FE\ 009A_H$ (CSIHnTX0W.CSIHnEDL = 0)

The following figure illustrates the timing.

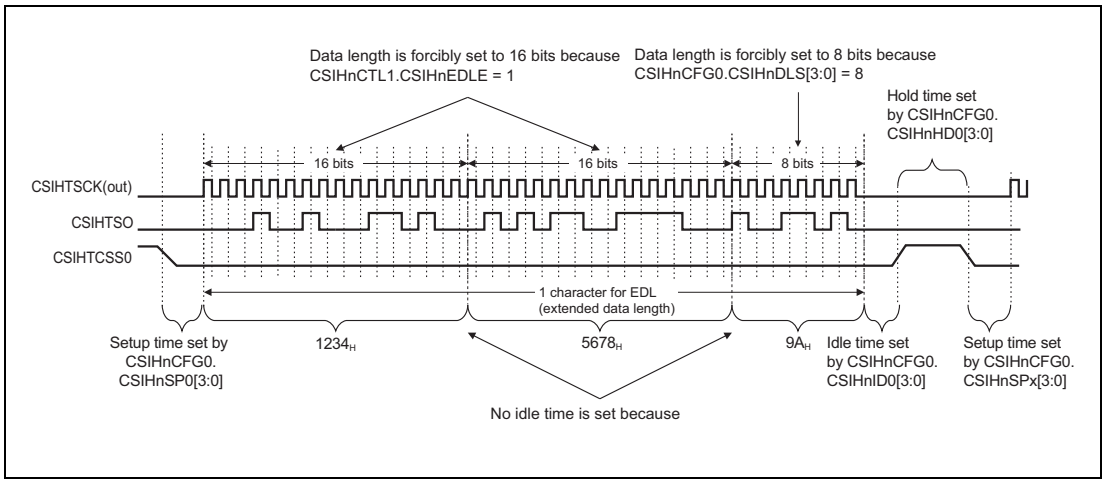



**Figure 16.25 EDL timing diagram**

**NOTES**

1. Data lengths setting of 1 bit is only permitted in combination with EDL mode.
2. It is not possible to transmit two consecutive data with a data length of 1 bit.
3. If parity is enabled, the parity bit is added after the last bit.
4. To consider the data direction, pay attention to the following example:
   - Data to be sent: $123456_H$
   - MSB first:
     Set CSIHnCFGx.CSIHnDIR = 0
     Write CSIHnTX0W = 2000 $1234_H$ (EDL bit = 1)
     Write CSIHnTX0W = 0000 $0056_H$ (EDL bit = 0)
   - LSB first:
     Set CSIHnCFGx.CSIHnDIR = 1
     Write CSIHnTX0W = 2000 $3456_H$ (EDL bit = 1)
     Write CSIHnTX0W = 0000 $0012_H$ (EDL bit = 0)
5. EDL mode cannot be used in receive-only mode of slave mode.
   ($CSIHnCTL1.CSIHnPRS[2:0] = 111_B$, CSIHnCTL0.CSIHnTXE = 0, CSIHnCTL0.CSIHnRXE = 1)
6. When setting less than 5bit of communication by DLS[3:0], Communication clock doesn't become active continuously,and communication wait sometimes occurs. When using by a slave mode, set the handshake function effective (CSIHnHSE = 1). Details are below

| | | **CSIHnDLS[3:0] settng** | | |
|---|---|---|---|---|
| | | **$0001_B$ 1 bit setting** | **$0010_B$ ~ $0101_B$ 2 bit ~ 5 bit setting** | **$0000_B$, $0110_B$~$1111_B$ 6 bit ~ 16 bit setting** |
| Communication bits <= 16 bit | Master | Setting prohibit | Communication wait sometimes occur | No limitation |
| | Slave | | CSIHnHSE=1 setting is needed | |
| Communication bits >= 17 bit | Master | Communication wait sometimes occur | Communication wait sometimes occur r | |
| | Slave | CSIHnHSE=1 setting is needed | CSIHnHSE=1 setting is needed | |

### 16.5.9 Serial data direction selection

The serial data direction is selectable for each chip select signal using the CSIHnDIRx bit in the CSIHnCFGx register.

The examples below show communication for a data length of 8 bit (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).

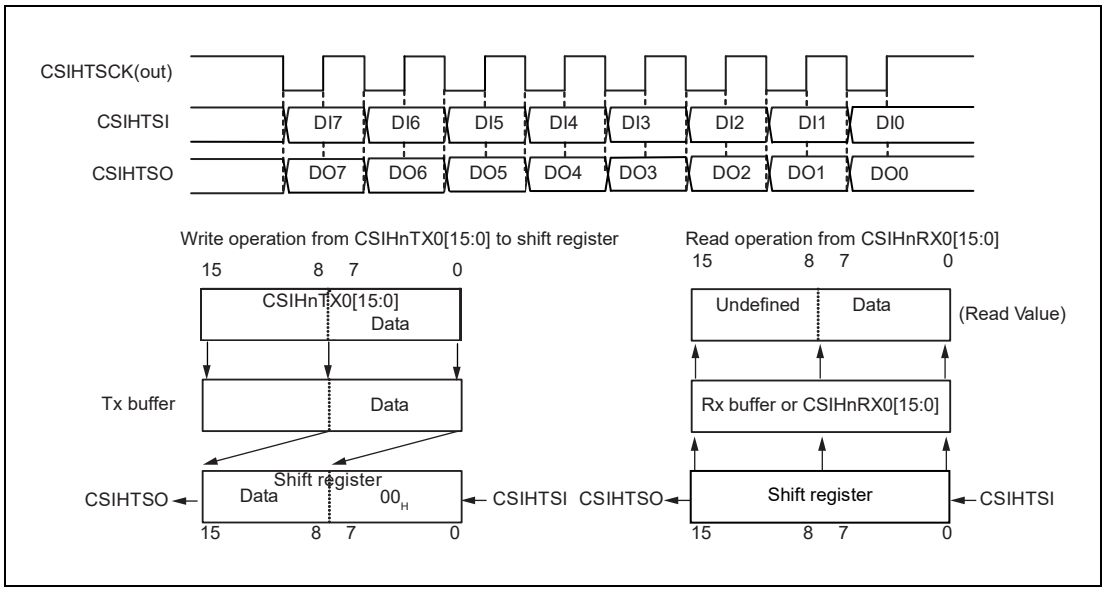



**Figure 16.26 Serial data direction select function - MSB first (CSIHnDIR = 0)**

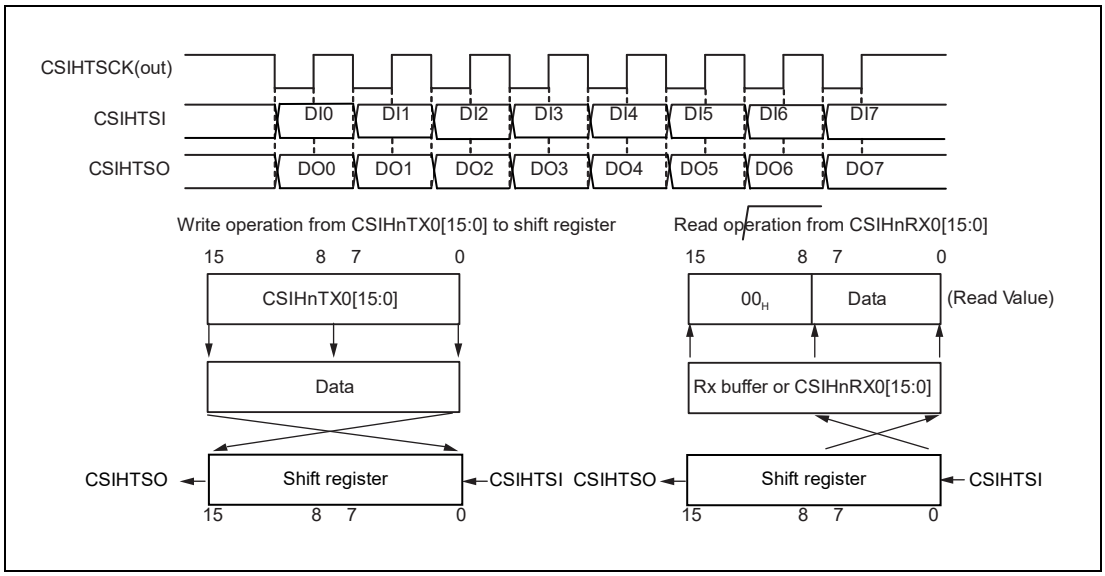



**Figure 16.27 Serial data direction select function - LSB first (CSIHnDIR = 1)**

## 16.5.10 SS Function

The SS function realizes communication between one master and multiple slaves (SPI communications).

In master mode, the master device outputs the salve select signal (CSIHTCSSx) to select a single slave. Communication by a device in slave mode is enabled when the slave input select signal (CSIHTSSI) is at the low level.

See the **Section 16.5.2, Master/slave connections**, for an example of a connection using the SS function.

### 16.5.10.1 SPI Communication Timing Using SS Function

The following figure illustrates the SPI communication signal using the SS function and timings.

In slave mode, the data transfer configuration is determined by the CSIHnCFG0 register.

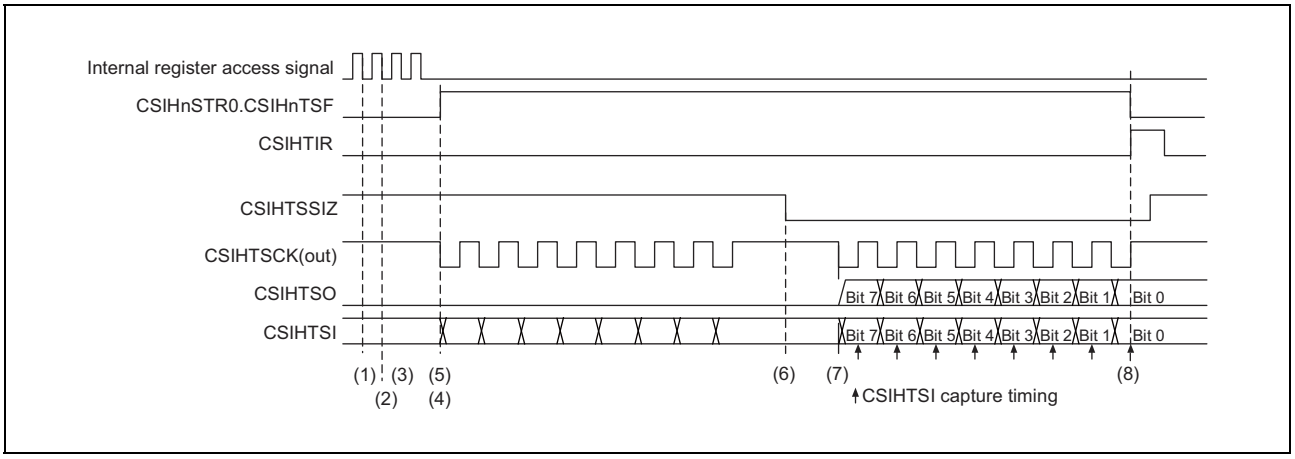



**Figure 16.28 Tx/Rx Timing of SPI Communication Using SS Function**

(1) CSIH is put into slave mode by setting CSIHnCTL2.CSIHnPRS[2:0] = $111_B$. CSIHnCFG0.CSIHnCKP0 and CSIHnCFG0.CSIHnDAP0 are 0.

(2) The data length is 8 bits (CSIHnCFG0.CSIHnDLS0[3:0] = $1000_B$). The data direction is MSB first (CSIHnCFG0.CSIHnDIR0 = 0).

(3) CSIH is set to transmit/receive operation mode (CSIHnCTL0.CSIHnTXE = 1, CSIHnCTL0.CSIHnRXE = 1). Communication start is permitted.

(4) The “transmission in progress” flag CSIHnSTR0.CSIHnTSF is automatically set when transfer data is written to the CSIHnTX0W or CSIHnTX0H transmission register.

**CAUTION**

**The TSF flag will be set with a certain delay after the set event (e.g. "Data is written to a transmit register"). Consider this behaviour when reading the status of this bit. Alternatively use the corresponding interrupt / interrupt status flag to monitor the transfer status.**

(5) As long as signal CSIHTSSIZ is high, transmission/reception is not started, even if an external transmission clock CSIHTSCK is applied. Input at CSIHTSI is ignored.

(6) As soon as CSIHTSSIZ falls to low level, CSIHTSO is enabled and ready for transmission.

(7) Now, as soon as the external clock signal CSIHTSCK appears, the slave transmits data to CSIHTSO and simultaneously captures data from CSIHTSI.

(8) Interrupt CSIHTIR indicates when the reception is complete. The CSIHnRX0W/H register can be read.The TSF flag will be cleared if data could be transferred from shift register to empty RX register. If RX register was not empty, TSF flag will remain high until RX register has been read.

### 16.5.10.2 CSIHTSSO operation

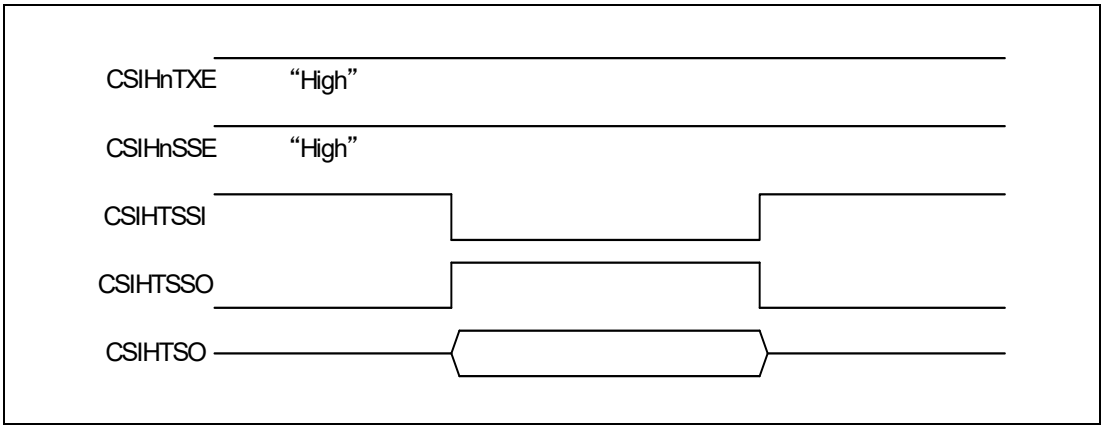
| CSIHnPWR | CSIHnTXE | CSIHnRXE | CSIHnSSE | CSIHTSSO |
|---|---|---|---|---|
| 0 | - | - | - | H |
| 1 | - | - | 0 | H |
| | 0 | | 1 | H |
| | 1 | | 1 | Reversed value of CSIHTSSI level |

CSIHTSSO pin is a signal to control I/O function of chip SO pin in case of using SS function.

CSIHTSO pin is enabled when CSIHTSSO pin is “High”. (Chip SO pin is drive.)

CSIHTSO pin is disabled when CSIHTSSO pin is “Low”. (Chip SO pin is not drive.)

(Operation of CSIHTSSO)




**CAUTION**

**If CSIHTSSI pin is changed during communication (CSIHnSTR0.CSIHnTSF = 1), current communication is not assured.**

### 16.5.11 Handshake function

CSIH features a handshake function to synchronize the master and the slave devices. This function can be enabled/disabled by bit CSIHnCTL1.CSIHnHSE. For handshake, the signals CSIHTRYI and CSIHTRYO are used.

The timing depends on the data phase selection bit, CSIHnCFGx.CSIHnDAPx setting.

#### 16.5.11.1 Slave mode

When CSIHnCTL1.CSIHnHSE = 1 and the slave is busy, the CSIHTRYO signal outputs low level (0). This can happen in two cases:

1. When memory mode is FIFO mode:
   The slave is in transmit-only or transmit/receive mode but has no transmission data in its buffer. This status is indicated by the flag, CSIHnSTR0.CSIHnEMF.

The example below is on the assumption of an eight-bit data length.

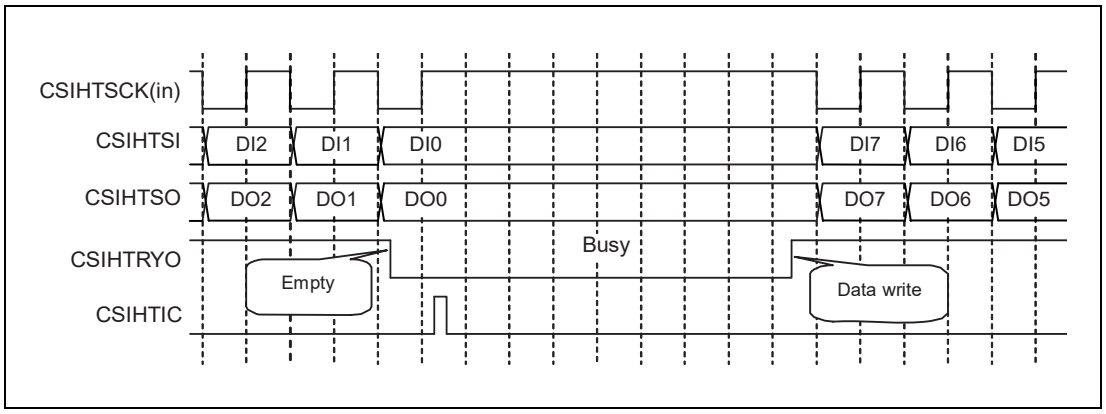



**Figure 16.29 Busy signal from the slave (FIFO mode; CSIHnCFGx.CSIHnDAPx = 0)**

The slave sets CSIHTRYO to high (“ready”) as soon as new transmission data is written to the FIFO.

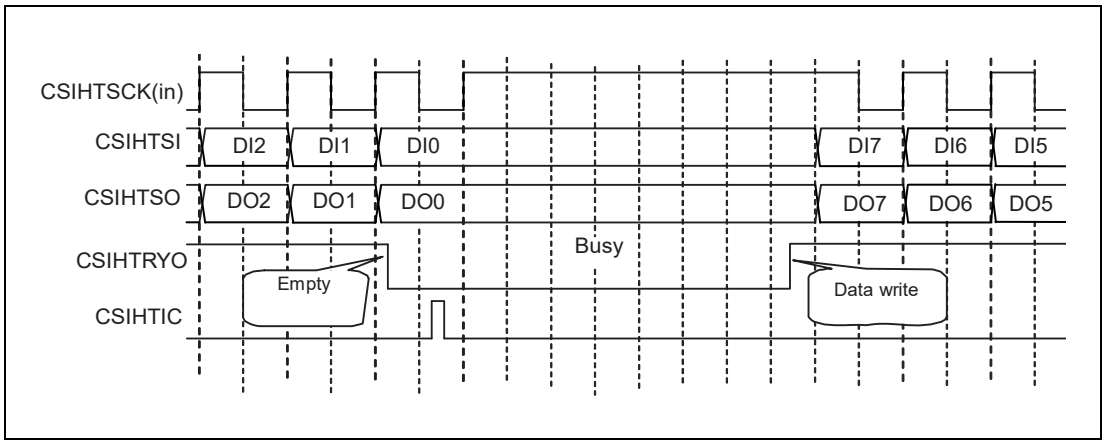



**Figure 16.30 Busy signal from the slave (FIFO mode; CSIHnCFGx.CSIHnDAPx = 1)**

2. When memory mode is direct access mode:
   Because the slave is in receive-only or transmit/receive mode but previously received data is still in the CSIHnRX0 register, new data cannot be copied from the shift register to CSIHnRX0 (CSIHnRX0 full condition).

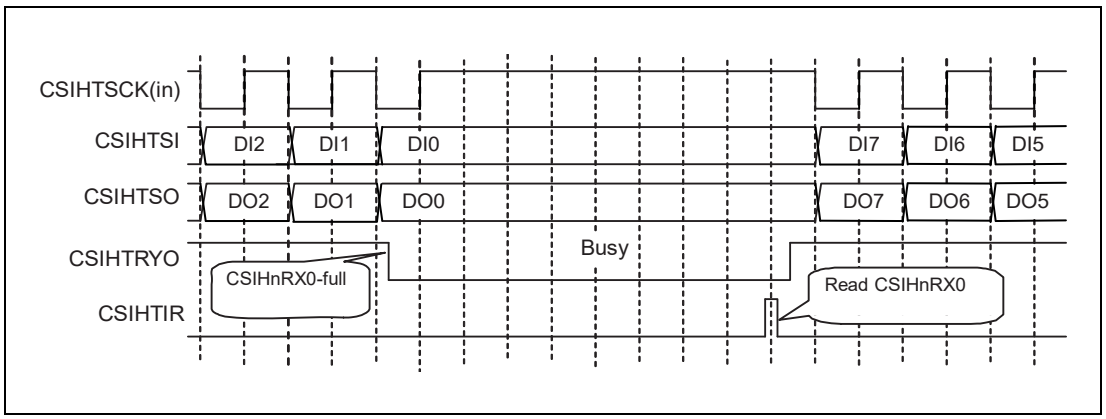



**Figure 16.31 Busy signal from the slave (Direct access mode; CSIHnCFGx.CSIHnDAPx = 0)**

The slave sets CSIHTRYO to high (“ready”) as soon as the reception register CSIHnRX0 has been read.

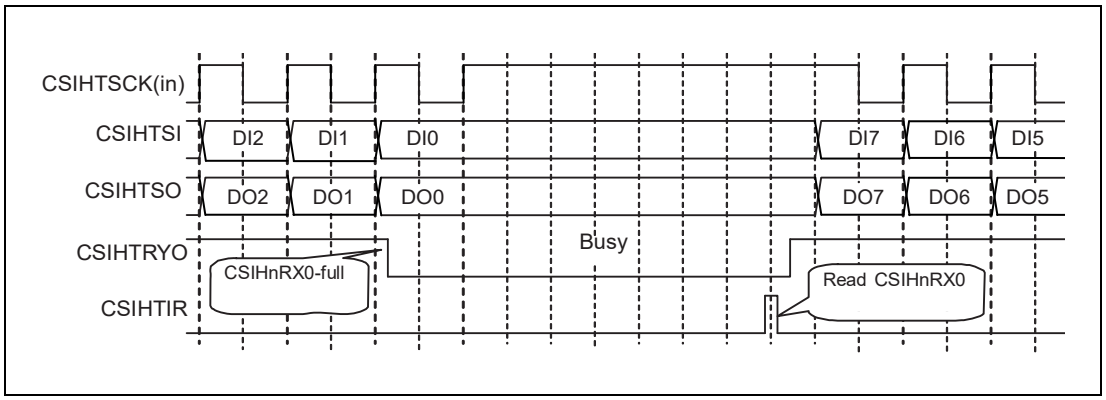



**Figure 16.32 Busy signal from the slave (CSIHnCFGx.CSIHnDAPx = 1)**

### 16.5.11.2 Master mode

When the master detects CSIHTRYI = 0, the following transfer is put on hold, and the master goes into wait status. It suspends the CSIHTSCK clock.

The CSIHTRYI level is checked at each half clock cycle of CSIHTSCK.

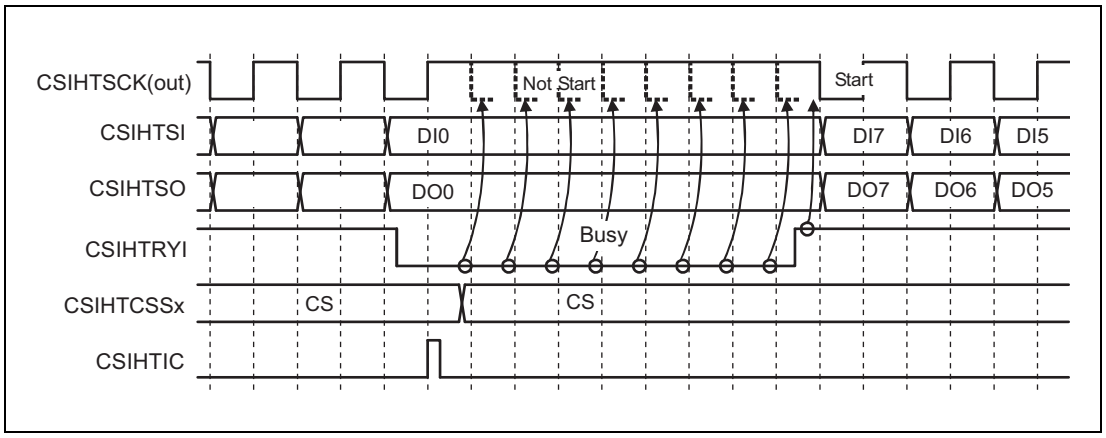



**Figure 16.33 Master's reaction on CSIHTRYI (CSIHnCFGx.CSIHnDAPx = 0)**

CSIHTRYO must be pulled down by the slave before the next transfer starts.

The master resumes the communication as soon as CSIHTRYO becomes high (the slave is "ready").

**NOTE**

CSIHTRYO output level is fixed to "Low" in master mode.

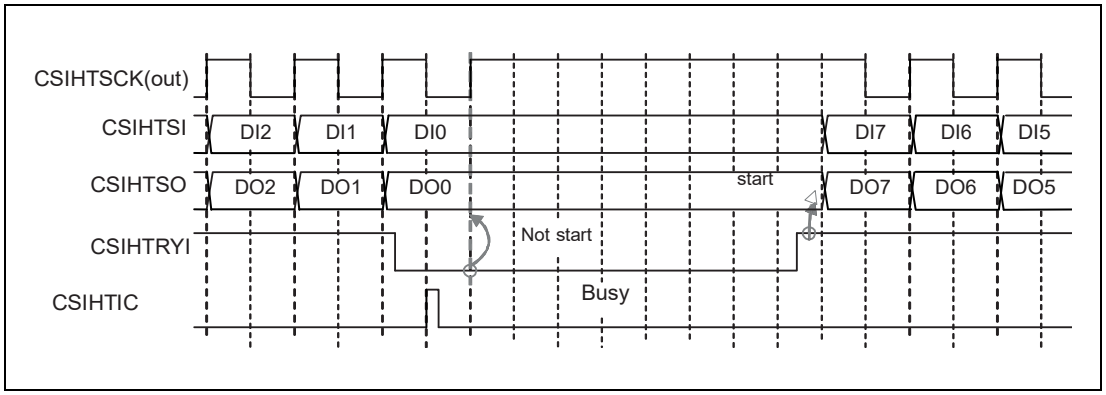



**Figure 16.34 Master's reaction on CSIHTRYI (CSIHnCFGx.CSIHnDAPx = 1)**

**CAUTION**

**If multiple slaves are connected, the master must only detect the CSIHTRYI signal of the slave it has selected for communication.**

**Even if CSIHTRYO signal comes from slave to master CSIHTRYI during transfer, communication doesn't wait until completion of data transfer.**

### 16.5.12 Error detection

- CSIH can detect five error types:
- Data consistency error (transmission data)
- Parity error (received data)
- Overrun error (received data)
- Time-out error (in FIFO mode)
- Overflow error (in FIFO mode)

Check for parity, data consistency and time-out errors can be enabled/disabled individually.

If one of these errors is detected, the interrupt request, CSIHTIRE is generated and the corresponding flags are set.

#### 16.5.12.1 Data consistency check

The purpose of the data consistency check is to ensure that the data physically sent as output signal is identical to the original data that was copied to the shift register.

The data consistency check can be enabled/disabled by bit CSIHnCTL1.CSIHnDCS. It is not active if data transmission is disabled (CSIHnCTL0.CSIHnTXE = 0).

When the data consistency check is active, the data transferred from CSIHnTX0W or CSIHnTX0H to the shift register is copied to a separate register. In addition, the physical levels at CSIHTSO are read back via the CSIHTDCS signal into an own shift register.

After completion of the transmission, the sent data is compared with the original transmission data.

Mismatch is considered as a data consistency error:

- Interrupt CSIHTIRE is generated.
- Bit CSIHnSTR0.CSIHnDCE is set.

Additionally, CSIHnRX0W.CSIHnTDCE of data that contains the error is set.

The data consistency check function is illustrated in the following block diagram.

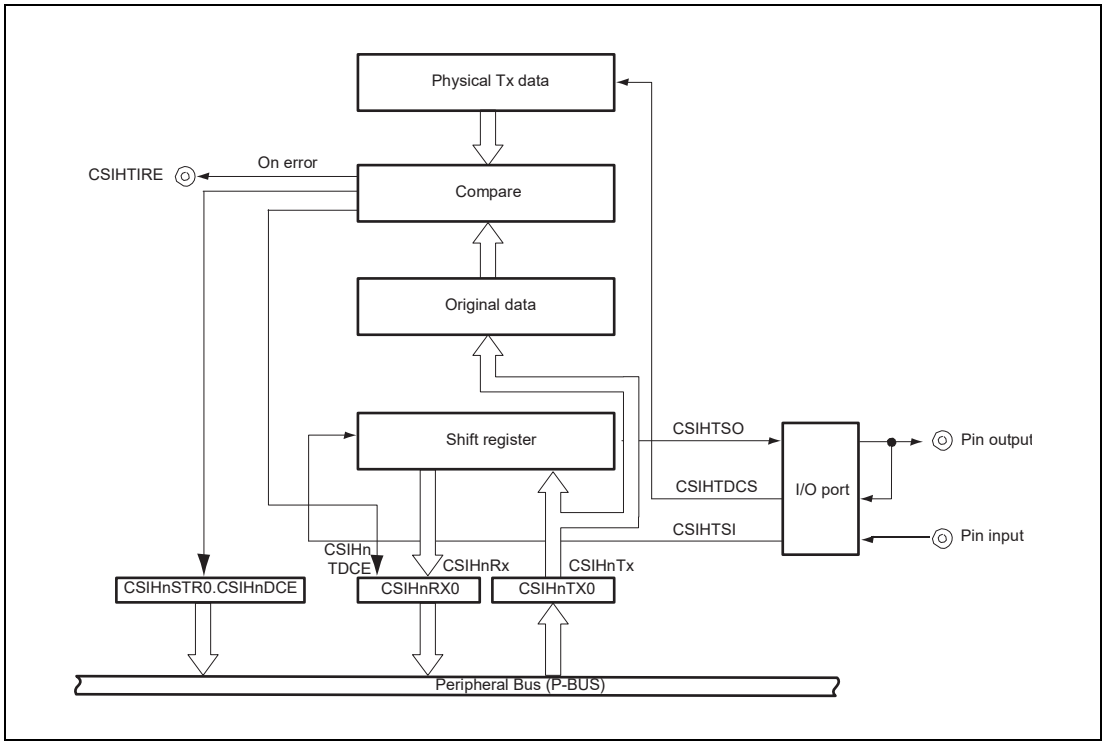



**Figure 16.35 Data consistency check functional block diagram**

### 16.5.12.2 Parity check

CSIH can append a parity bit to the last data bit (even if extended data length is used).

The use and type of parity is specified in CSIHnCFGx.CSIHnPSx[1:0].

Parity check is enabled if CSIHnCFGx.CSIHnPSx[1] = 1.

The parity bit is checked after a reception is complete. In case of parity error:

- Interrupt CSIHTIRE is generated.
- Bit CSIHnSTR0.CSIHnPE is set.

Additionally, CSIHnRX0W.CSIHnRPE of data that contains the error is set.

The figure below shows an example.

- Data length is 8 bits.
- The data to be transmitted is $05_H$ and $35_H$.
- Data direction is LSB first.
- Parity type is odd.

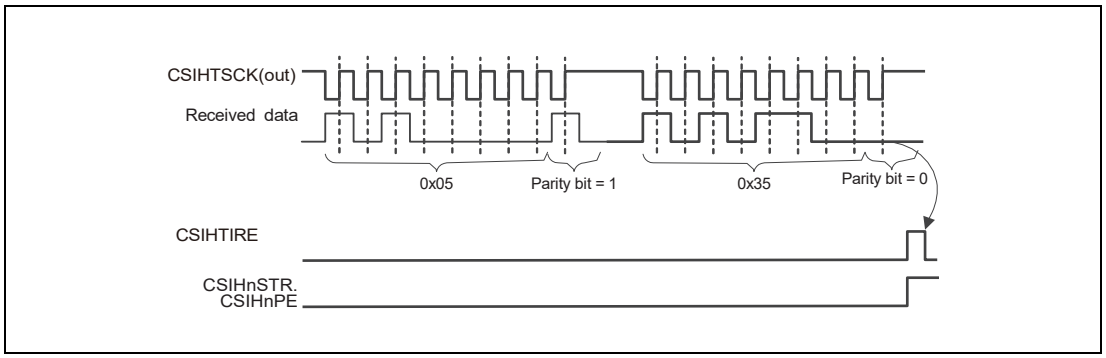



**Figure 16.36 Parity check example**

The parity bit of the first data is 1. There is no parity error, because the total number of ones (including the parity bit) is odd.

The parity bit of the second data is 0. This is detected as a parity error, because the total number of ones (including the parity bit) is even.

If the EDL (extended data length) function is used, the parity bit is added after the last bit of the data.

### 16.5.12.3 Time-out error

Time-out errors can be checked only in slave FIFO mode.

This error occurs if neither of the following occurred within a certain period of time:

- Received data in FIFO is read
- FIFO receives data from CSIHTSI

The time is defined in CSIHnMCTL0.CSIHnTO[4:0] in multiples of 8 times the transmission clock, CSIHSCK. A time-out error occurs when the specified time is exceeded (The time-out time is not detected when CSIHnMCTL0.CSIHnTO[4:0] = $00000_B$).

A dedicated time-out counter measures the time between the last and the next read operation.

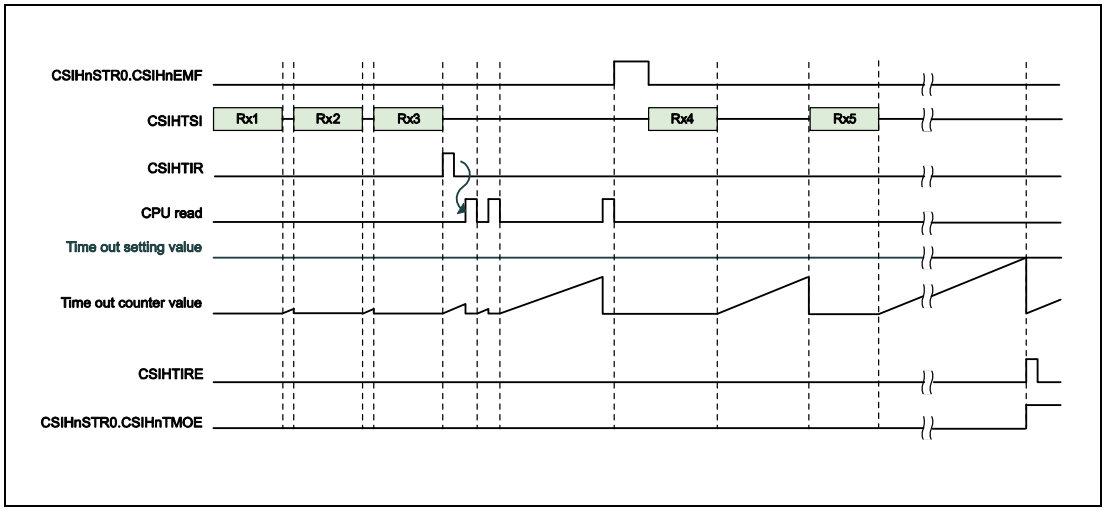



**Figure 16.37 Time-out check functional timing diagram**

The start timing of the time-out counter is as follows:

- When reception is completed
- When data read from the CPU completes
  (The counter does not start if the buffer is empty.)
- When a time-out error is detected

After a time-out error is detected, if data is still available in FIFO, the time-out counter restarts.

If the value set by bit CSIHnMCTL0.CSIHnTO[4:0] is reached again, the CSIHTIRE interrupt is output again.

The timeout counter continues to count until received data is read. To stop the counter, read all received data or set CSIHnSTCR0.CSIHnPCT to 1. Note that the pointer is cleared if you perform the latter.

The counter is reset at the following timing:

- Data is read once.
- New data is received.
- A timeout error is detected.
- The CSIHnSTCR0.CSIHnPCT bit is set to 1.

If a timeout error occurs, the following occurs:

- Interrupt CSIHTIRE is generated.
- Bit CSIHnSTR0.CSIHnTMOE is set.

### 16.5.12.4 Overflow error

An overflow error can happen in FIFO mode. It occurs when transmission data is written to the CSIHnTX0W register while the FIFO buffer is filled with received data.

#### Example

100 data packets have been transmitted. That means, the FIFO contains 100 received packets. The application starts to read the received data.

While the read operation is in progress, the application begins to write another set of 50 transmission data packets to the FIFO. However, only 10 received packets have been read up to now, 90 are still in the FIFO.

In this case, only 38 cells are available for new transmission data packets. When the CPU tries to write the 39th data packets, an overflow error happens.

This is illustrated in the following figure.

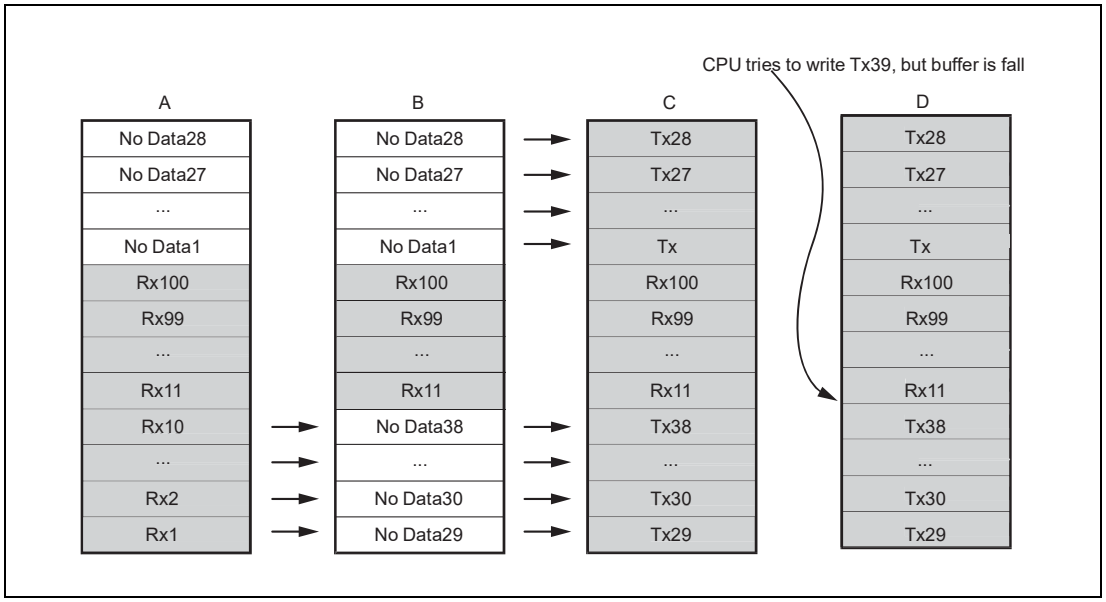



**Figure 16.38 FIFO overview**

The data packets after 39 are discarded. The figure below shows the overflow timing.

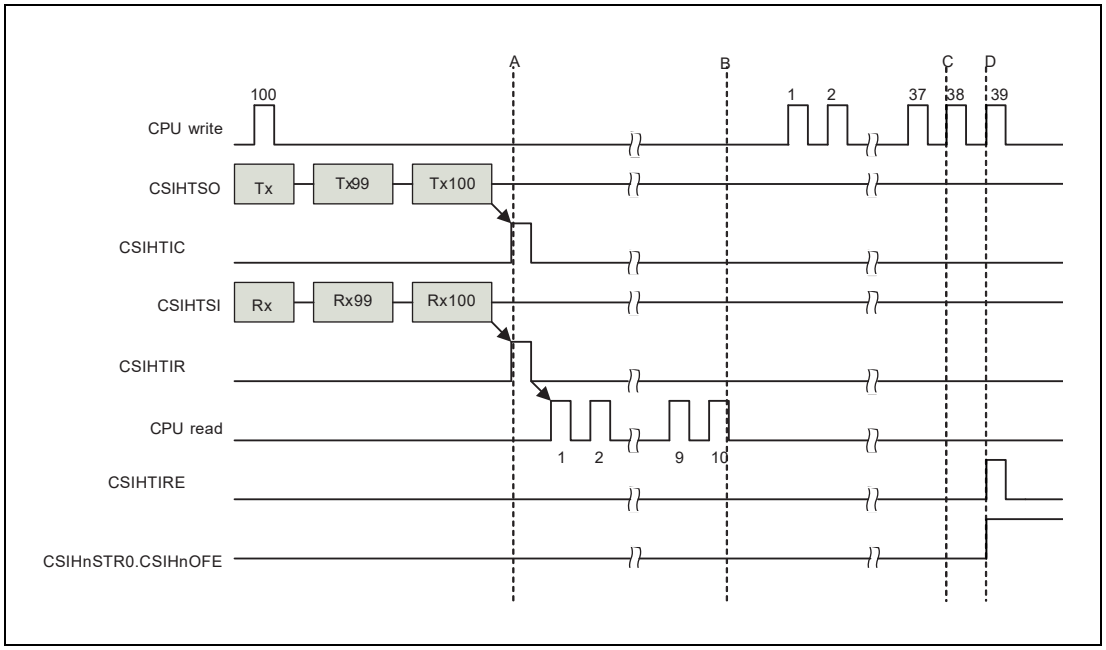



**Figure 16.39 FIFO overflow timing**

In case of overflow error:

- Interrupt CSIHTIRE is generated.
- Bit CSIHnSTR0.CSIHnOFE is set.

**CAUTION**

**If transmission data is written to CSIHnTX0W when buffer is full by reception data, overflow error is generated.**

### 16.5.12.5 Overrun error

An overrun error can happen in direct access, transmit-only buffer, and FIFO modes. It cannot happen in dual buffer mode. The overrun error is not generated if data reception is disabled (CSIHnCTL0.CSIHnRXE = 0).

#### (1) Direct access/transmit-only buffer

In direct access and transmit-only buffer mode, this error occurs when newly received data cannot be transferred from the shift register to the reception register CSIHnRX0. This happens when CSIHnRX0 was not read and therefore contains previous reception data.

The following figure illustrates the function.

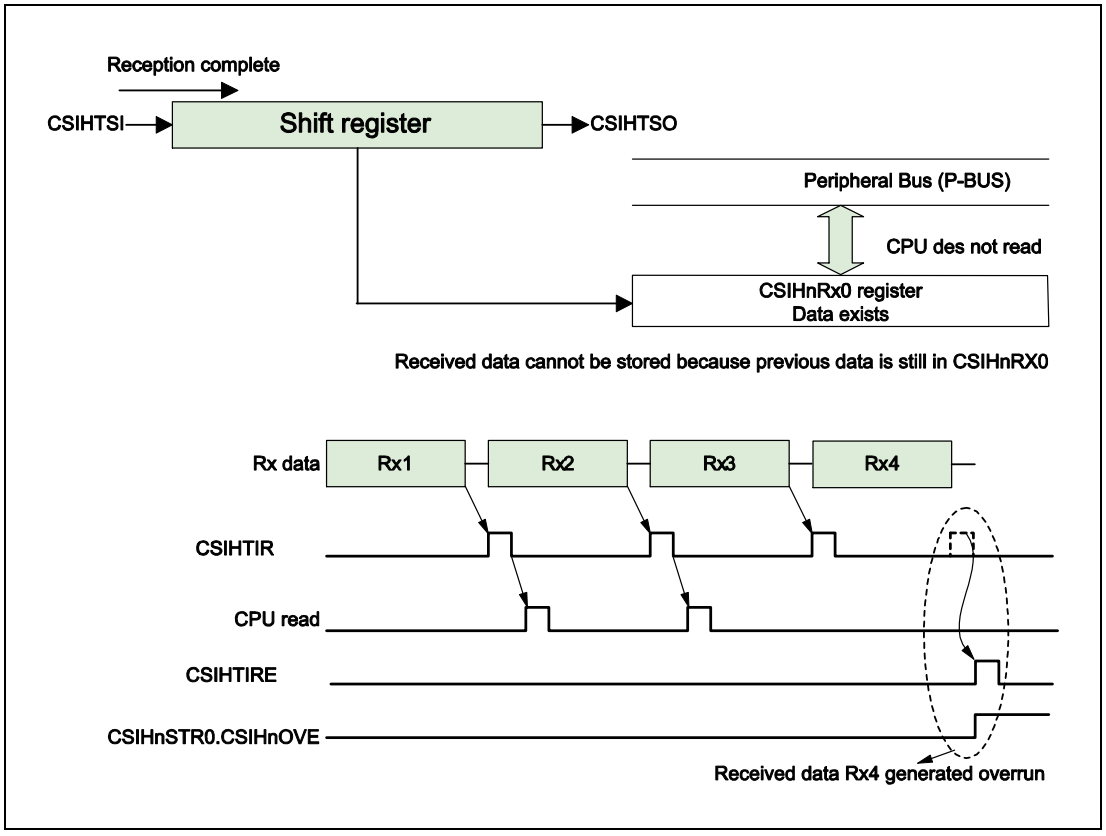



**Figure 16.40 Overrun error detection in direct access and transmit-only buffer mode**

**NOTE**

If overrun error is detected, INT_CSIHTIRE is generated instead of INT_CSIHTIR. CSIHnOVE bit of CSIHnSTR0 register is set to “1”. And reception data is overwritten to CSIHnRX0W.

### (2) FIFO mode

In FIFO mode, this error occurs if:

1. Newly received data cannot be transferred from the shift register to the FIFO because the FIFO is full.

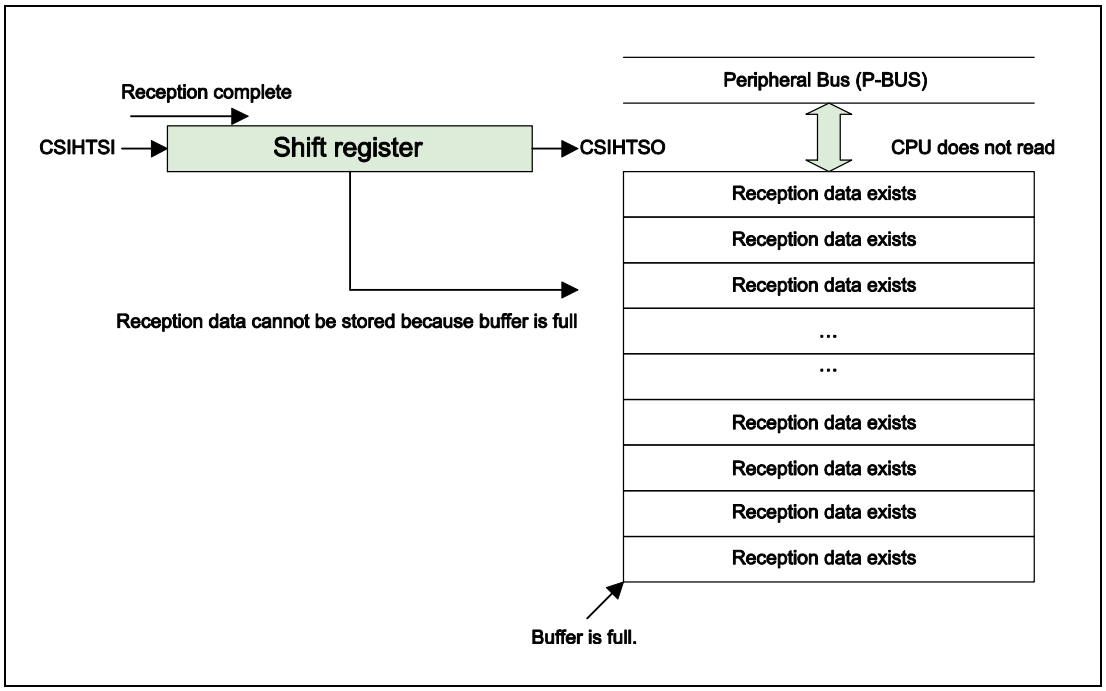



**Figure 16.41 Overrun error detection in FIFO mode (FIFO full)**

This error is detected only in FIFO and Receive mode. Reason is as follows.

- In FIFO and Transmit/Receive mode, transmission start is written to FIFO buffer.
- If Tx data is written in condition that FIFO buffer is full, overflow error is occurred.
- And that Tx data is not written to FIFO buffer, because buffer is full.
- Because transmission is not started. Therefore, overrun error is not occurred.

**NOTE**

If overrun error is detected, INT_CSIHTIRE is generated instead of INT_CSIHTIR. CSIHnOVE bit of CSIHnSTR0 register is set to “1”. And reception data is overwritten to CSIHnRX0W.

2. The CPU attempts to read non existing reception data.

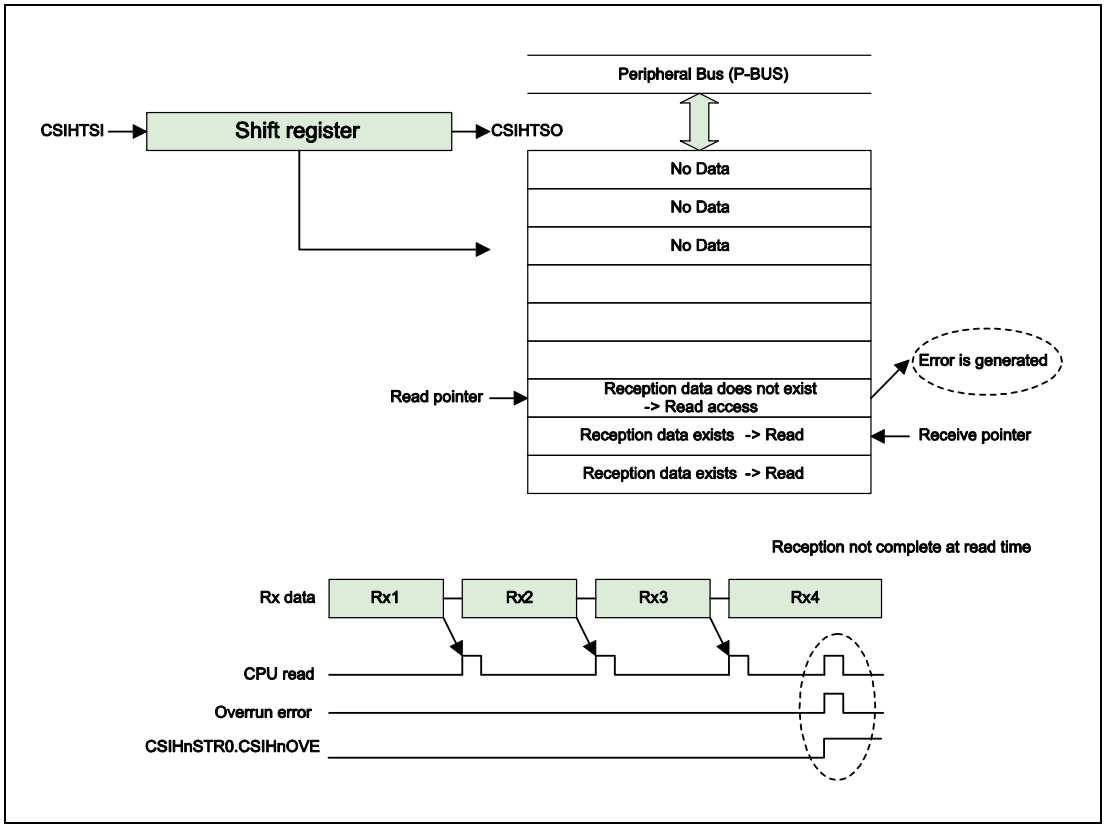



**Figure 16.42 Overrun error detection in FIFO mode (no data)**

3. In case of overrun error:
   - Interrupt CSIHTIRE is generated.
   - Bit CSIHnSTR0.CSIHnOVE is set.
   - Communication is stopped (except if the CPU tried to read non existing data).

**NOTE**

An overrun error can be avoided in slave mode by using the handshake function. When handshake is used in slave mode, the receiver (slave) signals to the transmitter (master) that it is busy. The transmitter then waits until the receiver has read its reception register and is ready again.

For details see **Section 16.5.11, Handshake function**.

### 16.5.13 Loop-back mode

Loop-back mode is a special mode for self-test. This feature is only available in master mode.

When this mode is active, the transmit and receive signals are internally connected, as shown in the figures below. The signals CSIHTSCK, CSIHTSO, and CSIHTSI are disconnected from the ports. In addition, the CSIHTSO output level is fixed to low, and CSIHTSCK is set to the inactive level as defined by CSIHnCFGx.CSIHnCKPx. The rest of CSIH works as in normal operation.

In order to test CSIH, put it in loop-back mode and carry out normal transfer operations. Then check that the received data is the same as the transmitted data.

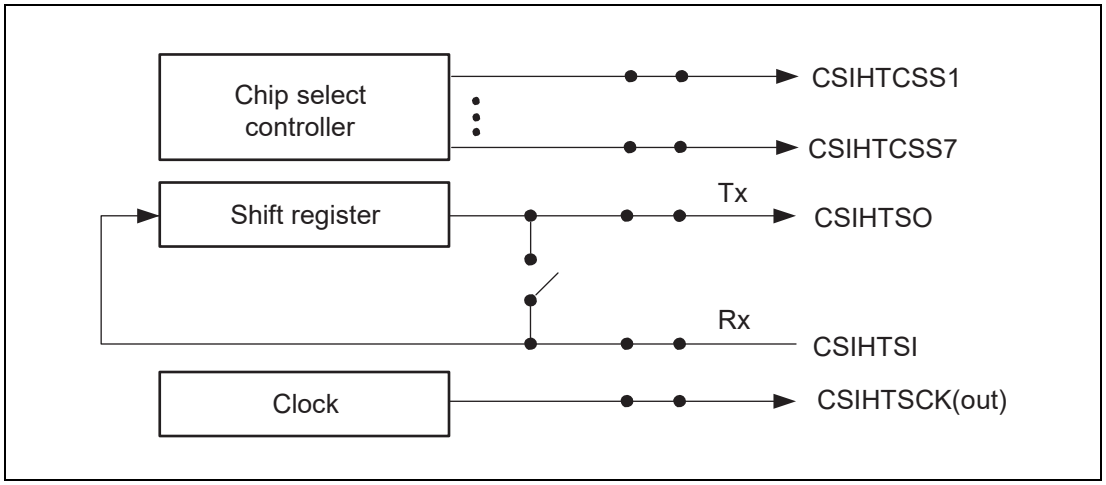



**Figure 16.43 Normal operation**

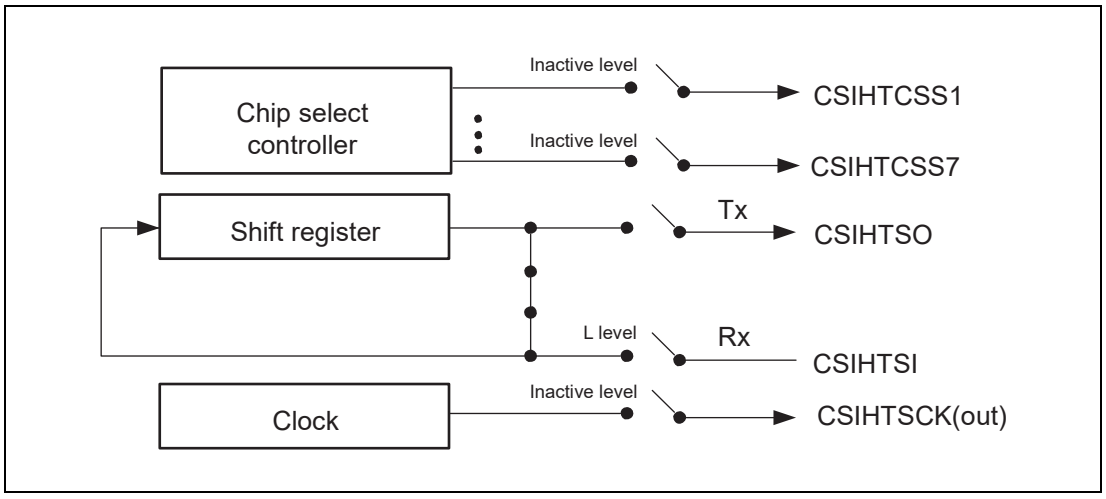



**Figure 16.44 Loop-back operation**

**CAUTION**

**When CSIHnCTL1.CSIHnLBM =1, macro operates as usual even if macro has received CSIHTRYI signal from slave.**

### 16.5.14 Enforced chip select idle setting

This macro is able to insert an idle state between the two consecutive transfer data by the setting of CSIHnCFGx.CSIHnIDLx. Detail is as follows.

1. When CSIHnCFGx.CSIHnIDLx = 0
   If a next CSIHTCSSx is the same as the previous one, an idle state is not inserted and an inter-data time is inserted.
   If a next CSIHTCSSx is different from the previous one, an idle state is inserted.
2. When CSIHnCFGx.CSIHnIDLx = 1
   An idle state is always inserted even if a next CSIHT is not different from the previous one.

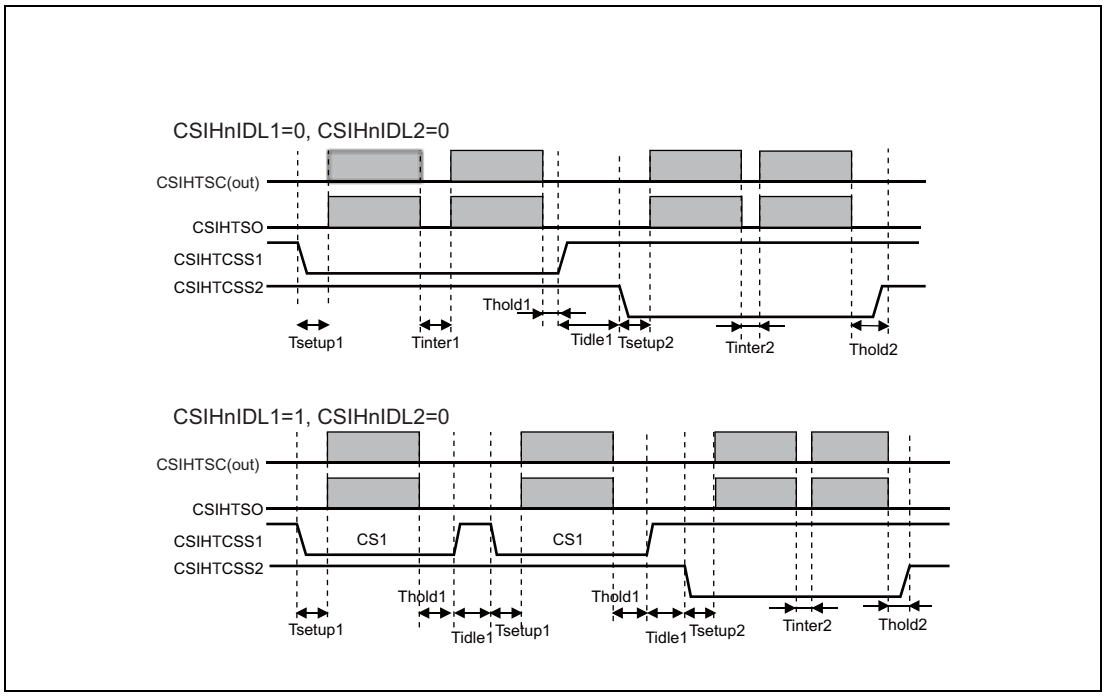



**Figure 16.45 Enforced Chip Select Idle Example**

### 16.5.15 Idle state control mode

CSIH is extended by a flexible idle state control mode.

In this mode the insertion of the idle state during the idle time can be controlled individually for each physical frame by the “idle state control bit” CSIHnISC in the upper 16 control bits of the transmit data.

In this mode, a couple of CSIHCSSn is used as chip select signal and control signal of communication direction. The configuration is defined as follows.

**Table 16.50 ISC mode CS-coupling**

| | Used as chip select | Used as transmit direction |
|---|---|---|
| Group0 | CSIHTCSS0 | CSIHTCSS1 |
| Group1 | CSIHTCSS2 | CSIHTCSS3 |
| Group2 | CSIHTCSS4 | CSIHTCSS5 |
| Group3 | CSIHTCSS6 | CSIHTCSS7 |

By setting the CSIHnISC bit, the behavior of the chip select signal (CS0, 2, 4 and 6) can be controlled.

If CSIHnISC is set, the CSIHTCSSn which is used as chip select will change to high after the end of the transmission.

If CHIHnISC is low, the CSIHTCSSn which used as chip select will not change to high state after the end of the transfer. Detailed timing is explained in **Table 16.51, Idle state control mode communication**.

**NOTES**

1. In the flexible idle state control mode the CSIHnIDLx bit has no function. Thereby, the mechanism of an automatically inserted idle state, if the chip select setting changes, is not available in this mode (CSIHnIDLx = 0).
2. In the flexible idle state control mode an active period of chip selection after the last data communication completion is maintained in CSIHnCTL1.CSIHnCSRI=0.
3. The change of the idle state control mode has no impact on the EDL mode.
4. In case of the idle state control enable (CSIHnISCE=1), when changing the configuration information (CSIHnCFG0-7) during each communication, CSIHnISC bit must be set to 1 (idle state insertion enable).
5. When CSIHnISC is set to 0, there is a possibility that slave device malfunctions because idle state is not inserted during communication.

**Table 16.51 Idle state control mode communication (1/3)**

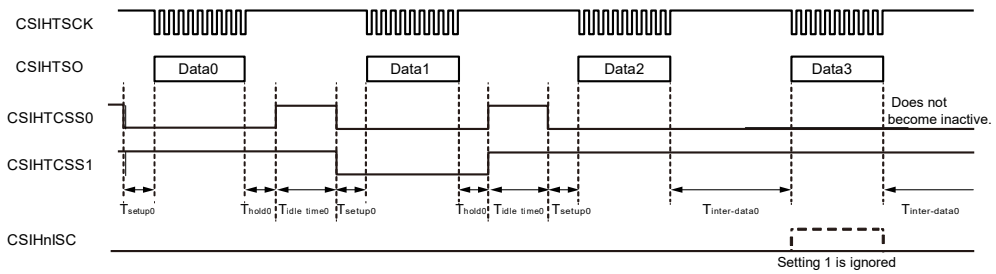
| CSIHn ISCE | CSIHn CSRI | CSIHn IDLx | Chip select output operation |
|---|---|---|---|
| 0 | 0 | 0 |  |

Table 16.51 Idle state control mode communication (2/3)

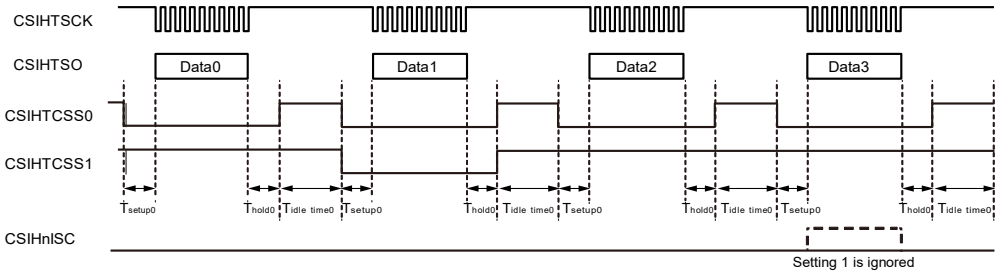
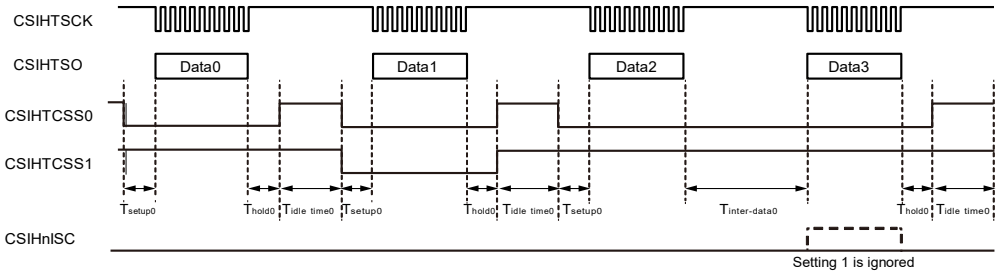
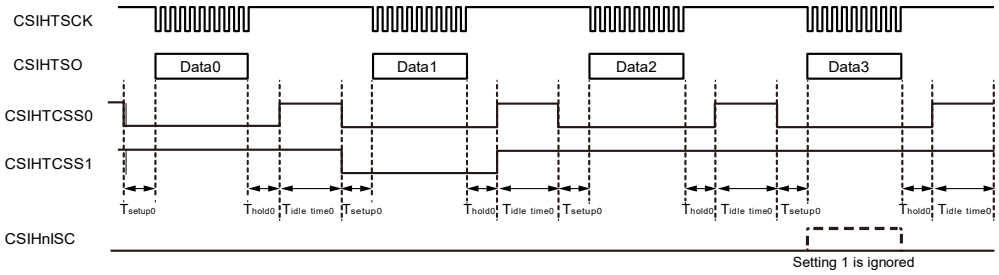
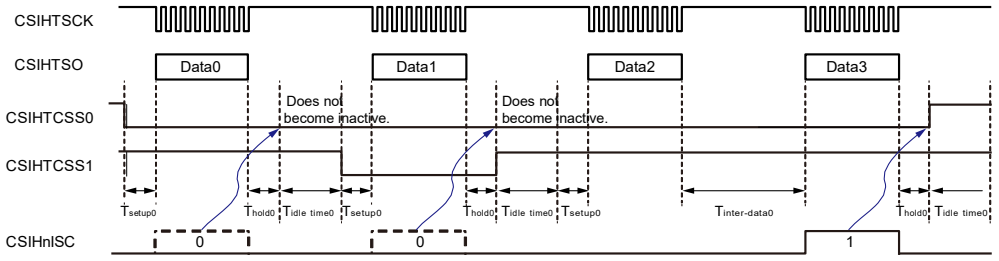
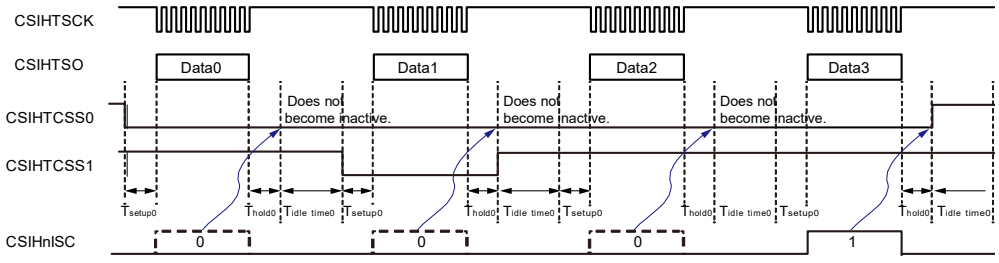
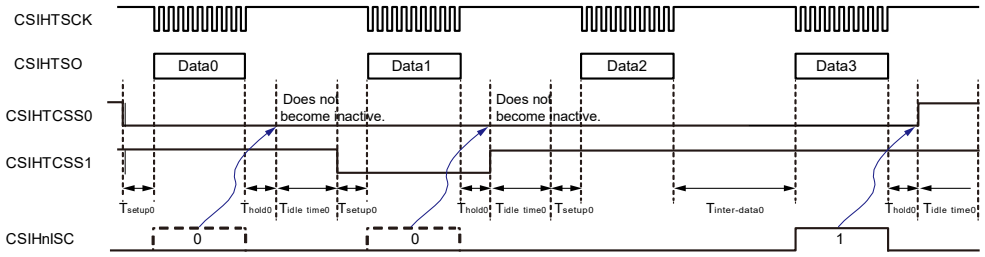
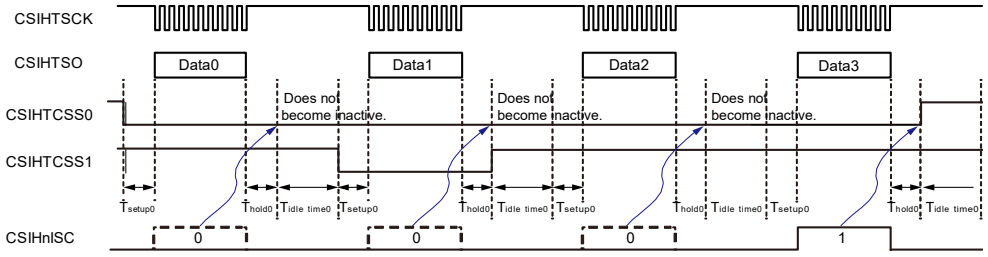
| CSIHn ISCE | CSIHn CSRI | CSIHn IDLx | Chip select output operation |
|---|---|---|---|
| 0 | 0 | 1 |  |
| 0 | 1 | 0 |  |
| 0 | 1 | 1 |  |
| 1 | 0 | 0 |  |
| 1 | 0 | 1 |  |

**Table 16.51 Idle state control mode communication (3/3)**

| CSIHn ISCE | CSIHn CSRI | CSIHn IDLx | Chip select output operation |
|---|---|---|---|
| 1 | 1 | 0 |  |
| 1 | 1 | 1 |  |

### 16.5.16 Silent mode

In silent mode, a “silent” communication can be inserted automatically after each physical frame. This silent communication will not generate a clock, data or chip select signal (CSIHTCSS[7:0] and CSIHSCK will become inactive; the data line will hold the end level of the former communication). This silent communication enables time slots with no communication. By using this, the customer can configure an extended idle time between two communications which cannot be interrupted by software. In each physical frame, the user can configure if 0-bit, 32-bit, 64-bit or 128-bit silent communication will be inserted afterwards.

The following table shows the possible silent communication time.

**Table 16.52 Silent time**

| Baud Rate (Mbps) | $T_{silent1}$ (µs) | | | |
|---|---|---|---|---|
| | 0-bit | 32-bit | 64-bit | 128-bit |
| 20 | 0 | 1.60 | 3.20 | 6.40 |
| 10 | 0 | 3.20 | 6.40 | 12.80 |
| 8 | 0 | 4.00 | 8.00 | 16.00 |
| 5 | 0 | 6.40 | 12.80 | 25.60 |
| 3.3 | 0 | 9.70 | 19.39 | 38.79 |

**Figure 16.44** shows a sample communication with extended idle time between two data transfers by silent mode.

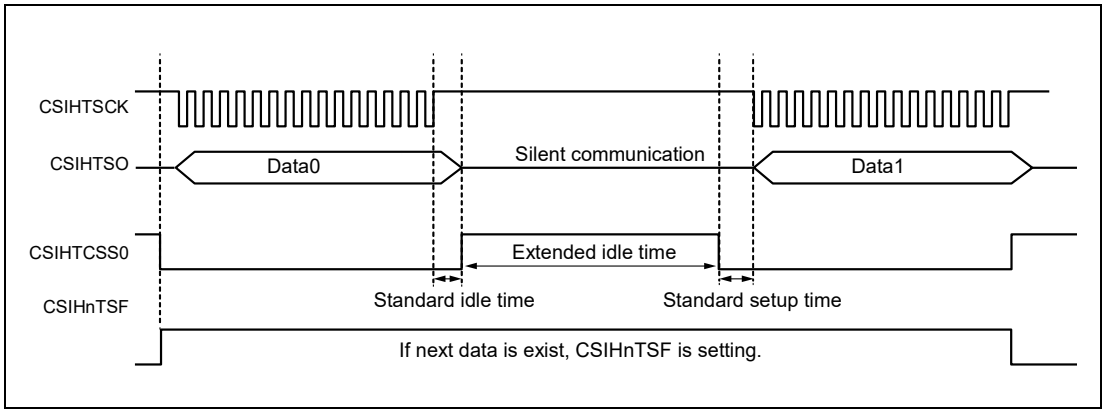



**Figure 16.46 Silent mode communication**

### 16.5.17 Limited Reset and Module stand-by

The CSIH can be reset by limited reset from SYSCTRL. The Limited Reset is functionally equivalent to a hardware reset. Software must ensure that CSIH is halted (CSIHnCTL0.CSIHnPWR bit = 0 and CSIHnCTL0.CSIHnTXE bit = 0 and CSIHnCTL0.CSIHnRXE bit =0 and PMMAnCTL.PMMAnPWR bit = 0)

The CSIH clock can be disabled by the SYSCTRL module stand-by function. Software must ensure that CSIH is halted if module stand-by enable.

### 16.5.18 Chip select Return to Inactive

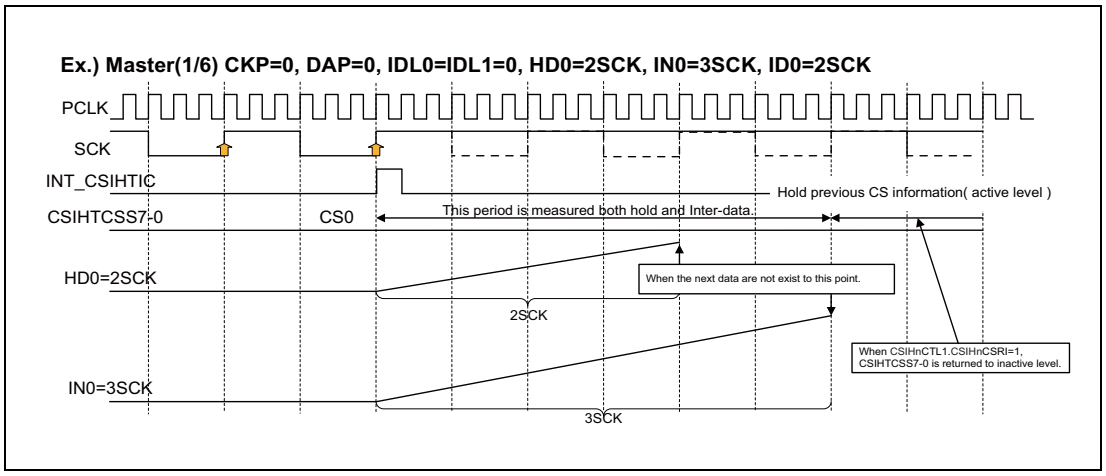
The CSIHnCTL1.CSIHnCSRI bit controls the level of CSIHTCSSn whether CS level is held active level or CS level is returned to inactive level after last transmission.**Note 1.**

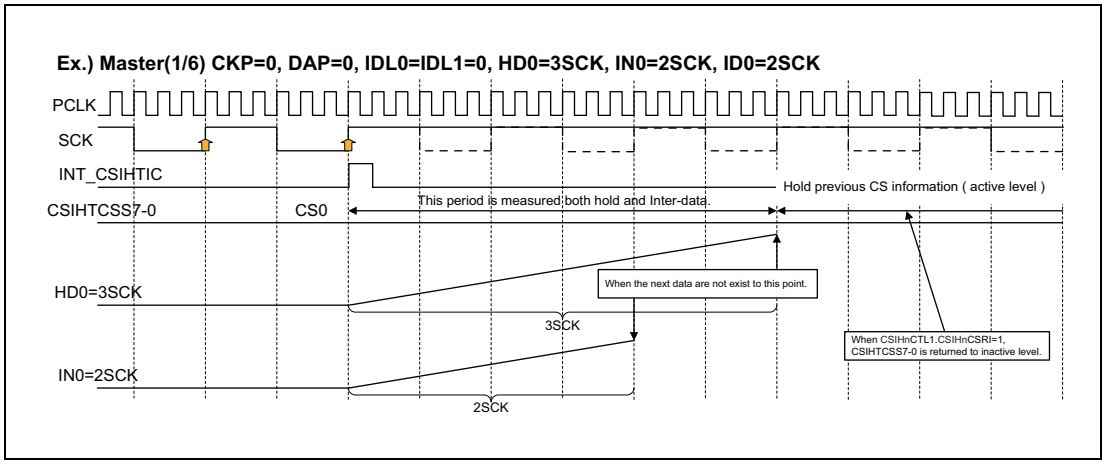
When CSIHnCTL1.CSIHnCSRI=0, CS level is held to active level after last transmission is completed.

When CSIHnCTL1.CSIHnCSRI=1, CS level is returned to inactive level after last transmission is completed.

When CSIHnCFG0-7.CSIHnIDLn=1, CS level during transfer is returned to inactive level.

**Note 1.** Chip select operation is as follows after last data is transmitted.
When CSIHnCFG0-7.CSIHnIDLn=0, CSIHnCTL1.CSIHnCSRI=0 and there is no next transmission data, macro operation is as follows.
Firstly, both inter-data and hold time are measured. If next transmission data is not prepared even macro has been waited for inter-data or hold time, CSIHTCSSn level is held to active level.




And, when the next transmission data is prepared during inter-data or hold time,

1. If previous CSIHnTX0W.CSIHnCS7-0 is as same as next CSIHnTX0W.CSIHnCS7-0,

   => 1) when the next transmission data is prepared before inter-data time, the next data willbetransferred after waiting for inter-data-time.

=> 2) when the next transmission data is prepared after inter-data time, the next data will be transferred by next operated clock.

2. If previous CSIHnTX0W.CSIHnCS7-0 is not same as next CSIHnTX0W.CSIHnCS7-0,

   => 1) when the next transmission data is prepared before hold-time, Idle-period is inserted after waiting for hold-time.

   => 2) when the next transmission data is prepared after hold-time, it's moving to Idle-period by next operated clock.

### 16.5.19 Select the Timing of Outputting INT_CSIHTIC

This function is used to select the timing of outputting CSIHTIC.

If CSIHnCTL1.CSIHnSLIT bit is set to 0, INT_CSIHTIC is generated after transfer.

If CSIHnCTL1.CSIHnSLIT bit is set to 1, INT_CSIHTIC is generated at start of transfer.

When CSIHnCTL1.CSIHnSLIT=0 and CSIHnCTL1.CSIHnSLIT=1, INT_CSIHTIC output timings are as follows. (Direct Access mode of CSIH-System.)

- In Master mode, INT_CSIHTIC is generated at point of 2PCLK from final edge of CSIHTSCO0-2 (regardless of DAP).

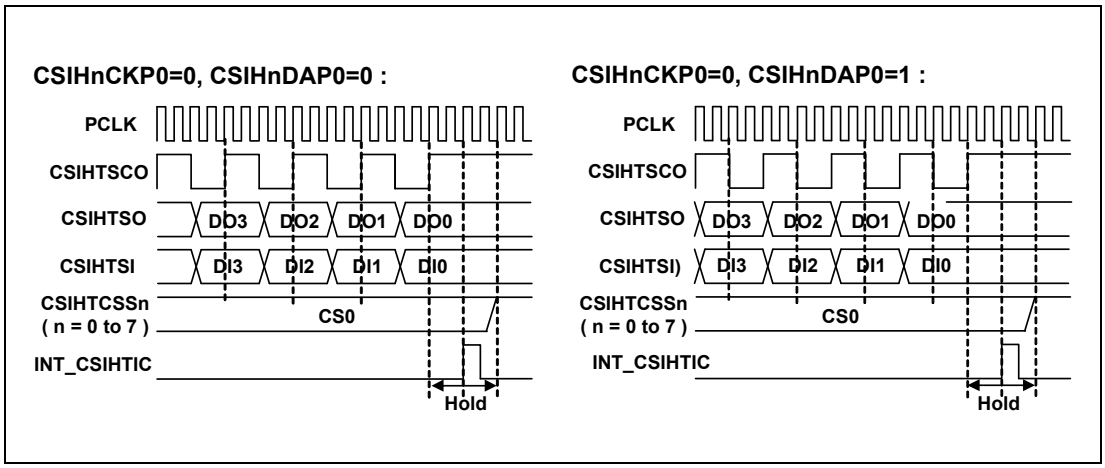



Figure 16.47 INT_CSIHTIC Output Timing in Master Mode (CSIHnCTL1.CSIHnSLIT = 0)

- In Slave mode, INT_CSIHTIC is generated at point of 2PCLK +α from final edge of CSIHTSCI (regardless of DAP).

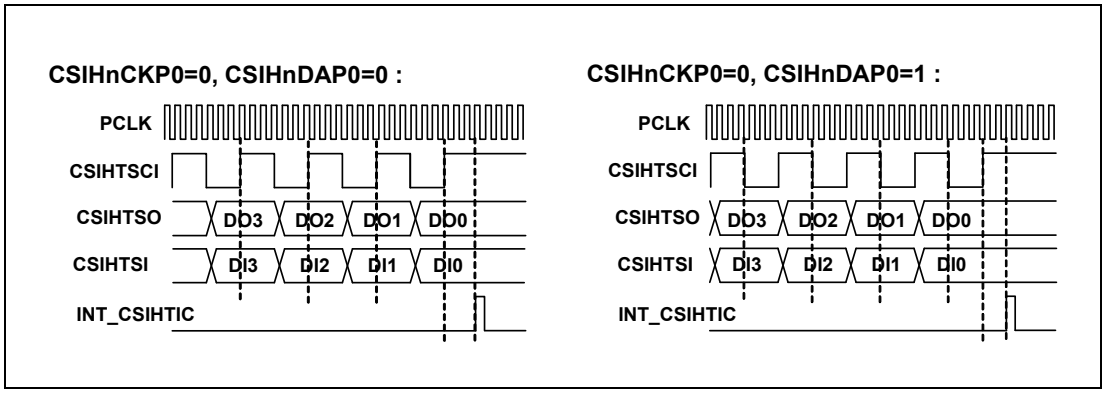



Figure 16.48 INT_CSIHTIC Output Timing in Slave Mode (CSIHnCTL1.CSIHnSLIT = 0)

- In Master mode, INT_CSIHTIC is generated at start edge of CSIHTSCO (regardless of DAP).

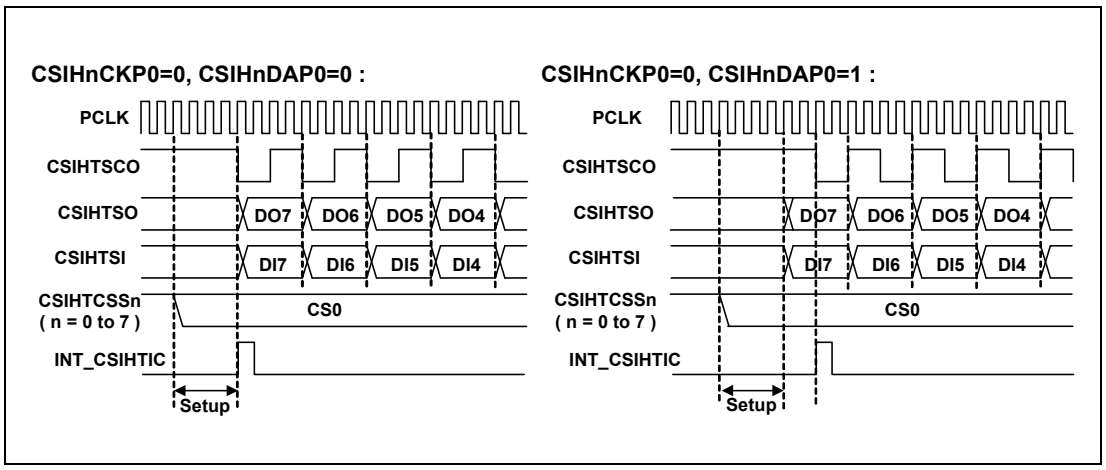



**Figure 16.49 INT_CSIHTIC Output Timing in Master Mode (CSIHnCTL1.CSIHnSLIT = 1)**

- In Slave mode, INT_CSIHTIC is generated at point of 2PCLK +α from start edge of CSIHTSCI (regardless of DAP).

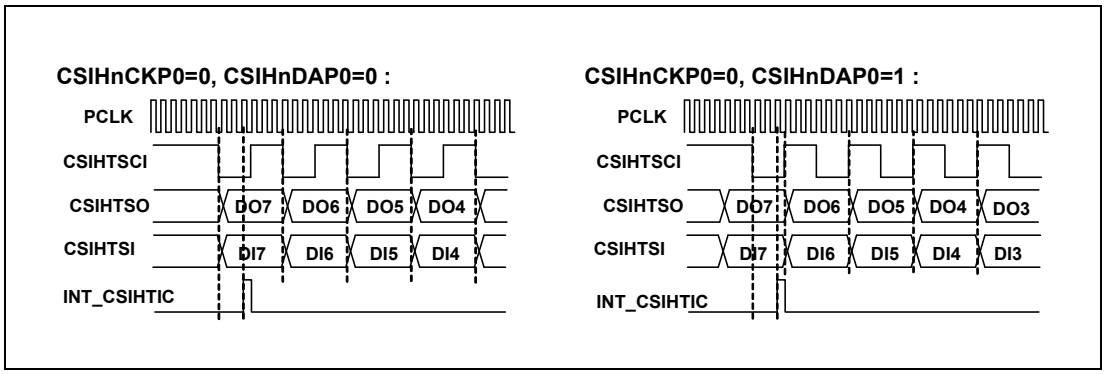



**Figure 16.50 INT_CSIHTIC Output Timing in Slave Mode (CSIHnCTL1.CSIHnSLIT = 1)**

This feature is only available in Direct Access mode of CSIH-System.

| Interrupt Name / Memory Mode | INT_CSIHTIC | INT_ CSIHTIR | INT_CSIHTIRE | INT_CSIHTIJC |
|---|---|---|---|---|
| Direct Access<br>• Tx Only Buffer (high priority communication is ongoing) | Enabled | Disabled | Disabled | Disabled |
| FIFO<br>• Tx Only Buffer (high priority communication is not ongoing)<br>• Dual Buffer | Disabled | Disabled | Disabled | Disabled |

## 16.6 Operating Procedures

The examples and procedures below are described according to the memory mode in the following order:

- Direct access mode
- Transmit-only buffer mode
- Dual buffer mode
- FIFO mode

**CAUTION**

**The behavior in the high priority communication of Tx Only Buffer mode is as same as the behavior in the Direct Access mode. The behavior in the high priority communication mode is illustrated in the section of Direct Access mode.**

### 16.6.1 Procedures in direct access mode

Two examples for a master are provided, one with job mode disabled, and the other one with job mode enabled.

#### 16.6.1.1 Transmit/receive in master mode when job mode is disabled

The procedures below is based on the assumption that:

- The transmission data length is 8-bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_{B)}$.
- Transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is disabled (CSIHnCTL1.CSIHnJE = 0).
- Normal CSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)
- Direct access mode (CSIHnCTL0.CSIHnMBS = 1)

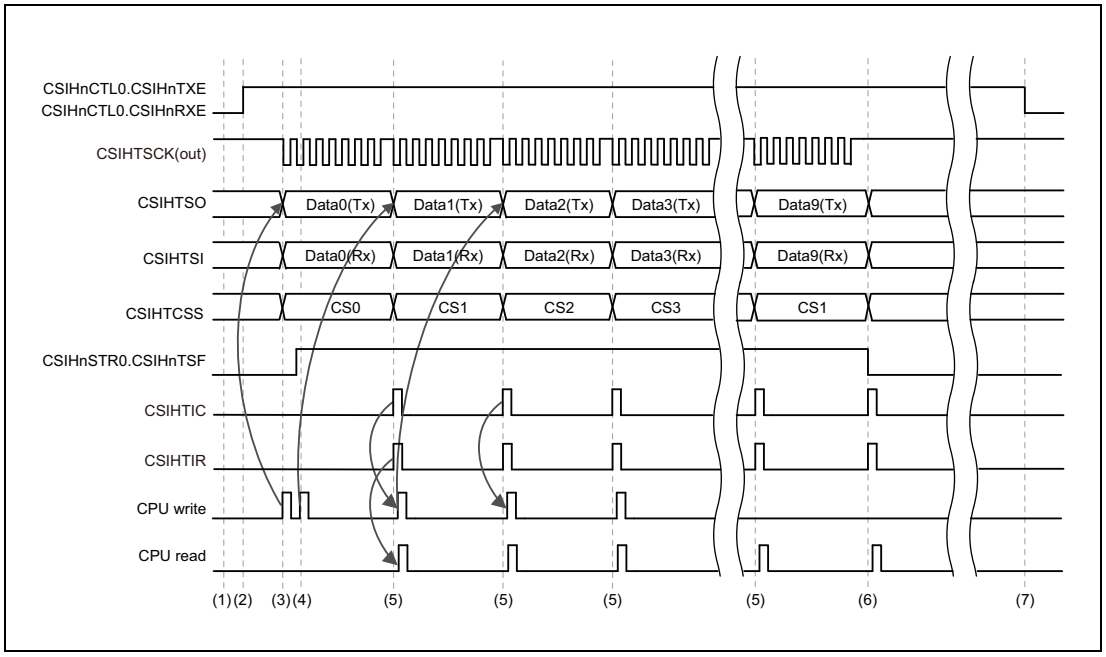



**Figure 16.51 Master in direct access mode, CSIHnCTL1.CSIHnJE = 0**

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. This example uses chip select signals CS0 to CS3.
   Specify the transfer mode and job mode by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
2. In the CSIHnCTL0 register, set bits CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), CSIHnRXE = 1 (permits reception).
   The output signal, CSIHTSO is enabled.
3. Write the first data packet to be sent to the transmission register, CSIHnTX0W. Within the same write operation, activate CS0. Transmission starts automatically when the first data becomes available.
4. Write the second data to CSIHnTX0W. If required, you can change the CS to address a different device. Writing the second data immediately after the first one avoids unnecessary delays between the packets.
5. After every transmission of a data packet the interrupts CSIHTIC and CSIHTIR are generated:
   - CSIHTIC indicates that the next data can be written to CSIHnTX0W.
   - CSIHTIR indicates that the reception register, CSIHnRX0 must be read.
6. No more write action is required after completion of packet 8. Packet 9 (the last packet) has been written in advance.
   However, reception register CSIHnRX0 must be read after completion of writing packets 8 and 9.
7. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation.

### 16.6.1.2 Transmit/receive in master mode when job mode is enabled

The procedure below is based on the assumption that:

- The transmission data length is 8-bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_{B)}$.
- Transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is enabled (CSIHnCTL1.CSIHnJE = 1).
- Normal CSIHTIC interrupt timing (CSIHnCTL1.CSIHnSLIT = 0)
- Direct access mode (CSIHnCTL0.CSIHnMBS = 1)
- Two jobs, each of them sends three data packets.

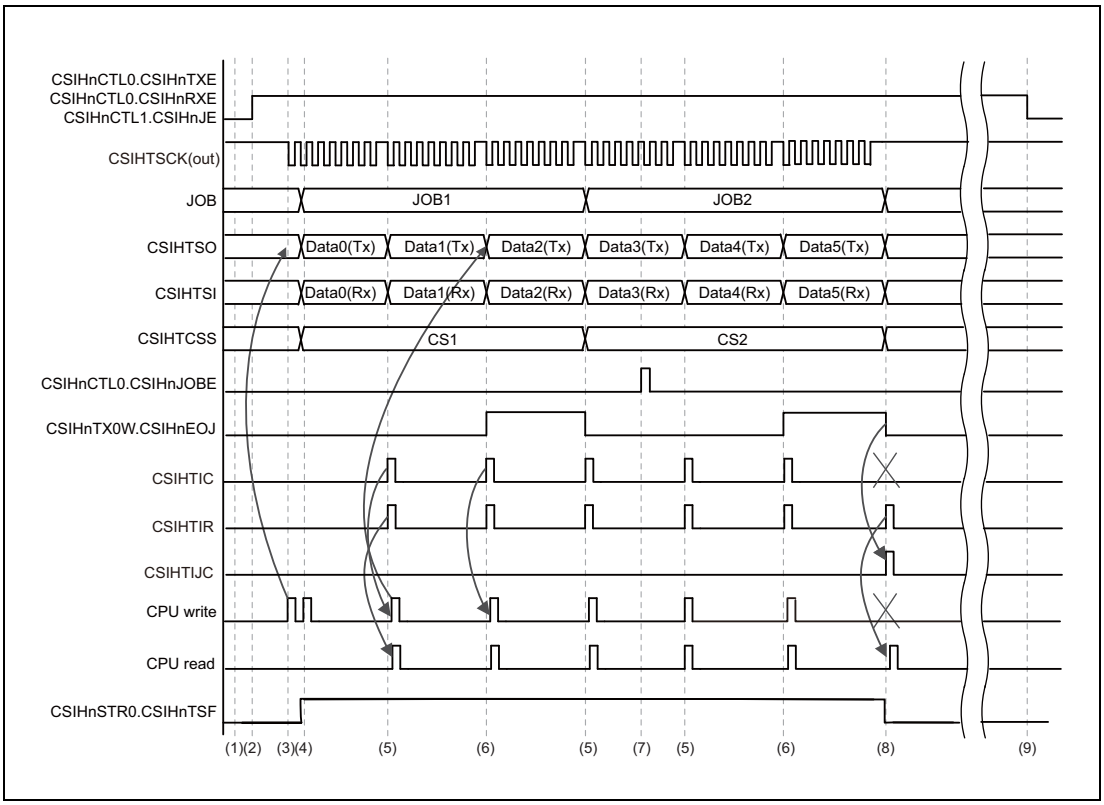



**Figure 16.52 Master in direct access mode, CSIHnCTL1.CSIHnJE = 1**

**CAUTION**

**"int_JOBE" in the above figure is internal signal of CSIHnCTL0.CSIHnJOBE bit.**

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. This example uses chip select signals CS1 and CS2.
   Specify the transfer mode and job mode by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
2. In the CSIHnCTL0 register, set the bits CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), CSIHnRXE = 1 (permits the reception), and CSIHnMBS = 1 (selects direct access mode).
   The output signal, CSHITSO is enabled.
3. Write the first data packet to be sent to the transmission register CSIHnTX0W. Transmission starts automatically when the first data becomes available.
   The CSIHnSTR0.CSIHnTSF flag indicates that communication is in progress.
4. Write the second data to CSIHnTX0H. Writing the second data immediately after the first one avoids unnecessary delays between the packets.
5. After every packet transmission, the interrupt requests, CSIHTIC and CSIHTIR are generated.
   - CSIHTIC indicates that the next packet can be written to CSIHnTX0.
   - CSIHTIR indicates that the reception register, CSIHnRX0 must be read.
6. Setting CSIHnTX0W.CSIHnEOJ = 1 indicates that the last data packet of the current job is sent. After that, the next job may begin.
7. By setting CSIHnCTL0.CSIHnJOBE = 1, communication is forced to stop at the end of the current job (JOB2).
8. After the forced stop of communication, the interrupt request, CSIHTIC is replaced by CSIHTIJC. CSIHTIR is generated as usual.
   The interrupt request, CSIHTIJC indicates a forced stop of communication at the end of the current job.
   The interrupt request, CSIHTIC is not generated. Additionally, the transmission data available in the CSIHnTX0 register is not sent.
9. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation.
   To start another transmission without stopping communication, perform steps 3 and later.

### 16.6.2 Procedures in transmit-only buffer mode

Two examples for a master is provided, one with job mode disabled, and the other one with job mode enabled.

#### 16.6.2.1 Transmit/receive in master mode when job mode is disabled

The procedure below is based on the assumption that:

- The transmission data length is 8-bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_{B)}$.
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is disabled (CSIHnCTL1.CSIHnJE = 0).
- The number of data packets is 9 (CSIHnMCTL2.CSIHnND[7:0] = $09_H$).
- The transfer start address is $10_H$ (CSIHnMCTL2.CSIHnSOP[6:0] = $10_H$).

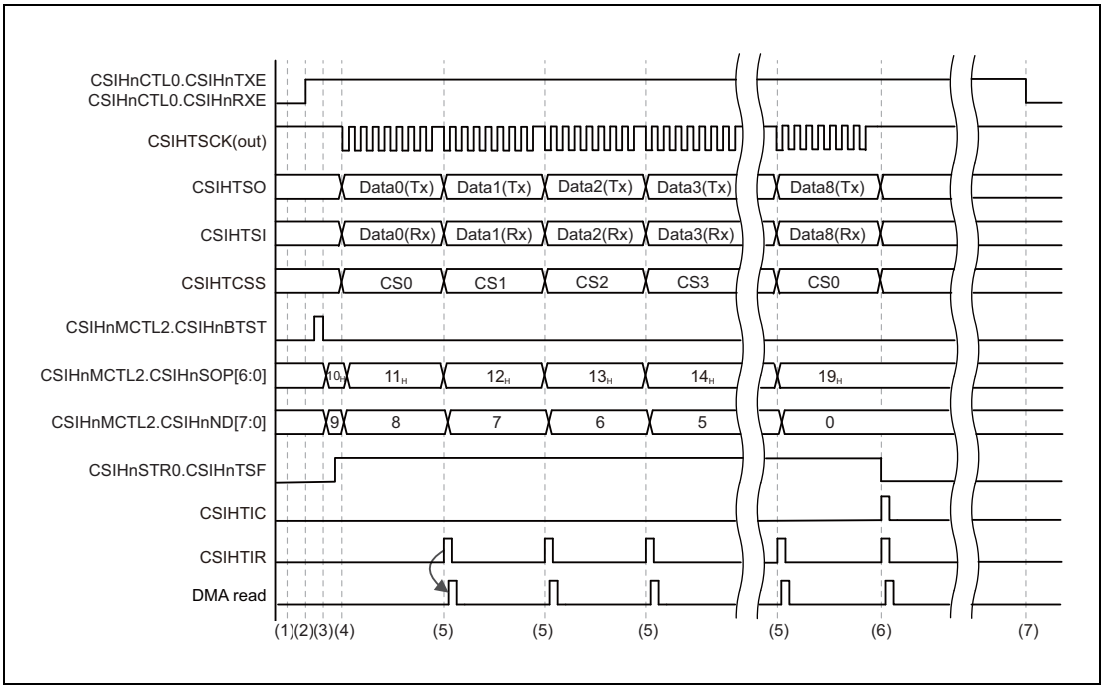



**Figure 16.53 Master in transmit-only buffer mode, CSIHnCTL1.CSIHnJE = 0**

**NOTE**

The procedure of writing the data into the buffer is not described.

Set CSIHnMRWP0.CSIHnTRWA6-0 to the first data address and write the transmit data to CSIHnTX0W. The CSIHnMRWP0.CSIHnTRWA6-0 is automatically incremented after each write.

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. This example uses chip select signals CS0 to CS3.
   Specify the transfer and operating modes by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
2. Configure the memory mode in CSIHnMCTL0.CSIHnMMS[1:0]. Set CSIHnMCTL0.CSIHnMMS[1:0] = $10_B$.
3. In the CSIHnCTL0 register, set the bits CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission) and CSIHnRXE = 1 (permits reception). The CSIHnCTL0.CSIHnMBS bit must be cleared.
4. Configure the send pointer and the number of data packets by setting bits CSIHnMCTL2.CSIHnSOP[6:0] and CSIHnMCTL2.CSIHnND[7:0]. Start the buffer transfer by setting CSIHnMCTL2.CSIHnBTST.
   Bit CSIHnMCTL2.CSIHnSOP[6:0] is automatically incremented and bit CSIHnMCTL2.CSIHnND[7:0] is decremented after each data packet transmission. Transmission/reception is started.
5. After every data packet transmission, the interrupt request, CSIHTIR is generated. CSIHTIR indicates that the reception register, CSIHnRX0 must be read.
6. When all transmissions are complete, the interrupt request, CSIHTIC is generated.
7. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation.

### 16.6.2.2 Transmit/receive in master mode when job mode is enabled

The procedure below is based on the assumption that:

- The transmission data length is 8-bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is enabled (CSIHnCTL1.CSIHnJE = 1).
- The number of data packets is 8 (CSIHnMCTL2.CSIHnND[7:0] = $08_H$).
- The transfer start address is $10_H$ (CSIHnMCTL2.CSIHnSOP[6:0] = $10_H$).

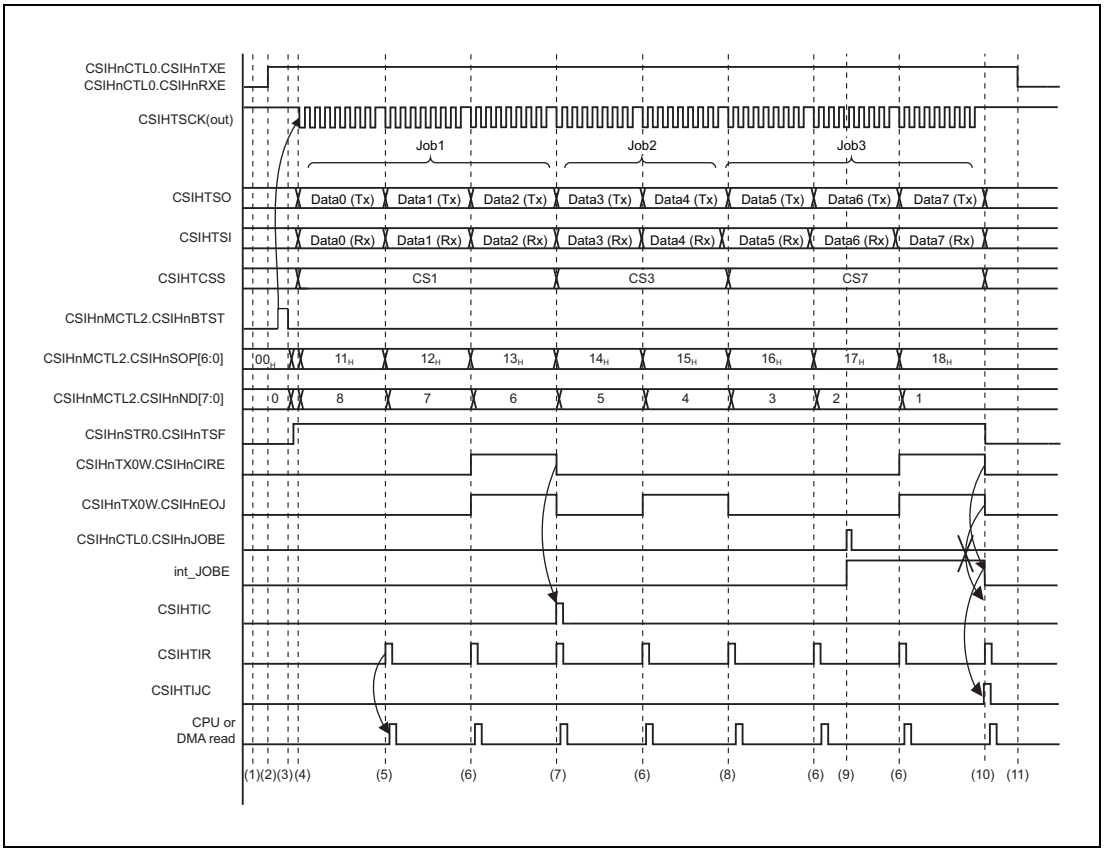



**Figure 16.54 Master in transmit-only buffer mode, CSIHnCTL1.CSIHnJE = 1**

**CAUTION**

The process of writing the data into the buffer is not described.

Set CSIHnMRWP0.CSIHnTRWA6-0 to the first data address and write the transmit data to CSIHnTX0W. The CSIHnMRWP0.CSIHnTRWA6-0 is automatically incremented after each write.

The int_JOBE signal in the timing chart above is an internal signal for the CSIHnJOBE bit.

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register.This example uses chip select signals CS1, CS3, and CS7.
   Specify the transfer mode and job mode by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
2. Configure the memory mode in CSIHnMCTL0.CSIHnMMS[1:0]. Set CSIHnMCTL0.CSIHnMMS[1:0] = $10_B$.
3. In the CSIHnCTL0 register, set bits CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), and CSIHnRXE = 1 (permits reception). The CSIHnCTL0.CSIHnMBS bit must be cleared.
4. Configure the send pointer and the number of data packets by setting bits CSIHnMCTL2.CSIHnSOP[6:0] and CSIHnMCTL2.CSIHnND[7:0]. Start the buffer transfer by setting CSIHnMCTL2.CSIHnBTST.
   Transmission is started. Bit CSIHnMCTL2.CSIHnSOP[6:0] is automatically incremented and bit CSIHnMCTL2.CSIHnND[7:0] is decremented after each data packet transmission.
5. After every data packet transmission, the interrupt request, CSIHTIR is generated. CSIHTIR indicates that the reception register, CSIHnRX0 must be read.
6. The CSIHnTX0W.CSIHnEOJ = 1 setting indicates that the last data of the current job is sent.
7. The interrupt request CSIHTIC is generated. CSIHTIC indicates that the last data of the current job (CSIHnTX0W.CSIHnEOJ = 1) was sent with CSIHnTX0W.CSIHnCIRE = 1.
8. The CSIHTIC interrupt request is not generated because the last data of the current job (CSIHnTX0W.CHABnEOJ = 1) was sent with CSIHnTX0W.CSIHnCIRE = 0.
9. By setting CSIHnCTL0.CSIHnJOBE = 1, communication is forced to stop at the end of JOB3.
10. After the forced stop of communication, interrupt requests CSIHTIJC and CSIHTIR are generated at the end of job3.
    The CSIHTIJC interrupt request indicates a forced stop of communication at the end of the current job.
    The CSIHTIC interrupt request is not generated because the CSIHTIJC interrupt request is generated instead of the CSIHTIC interrupt request. Additionally, the transmission data available in the CSIHnTX0 register is not sent.
11. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation.

### 16.6.3 Procedures in dual buffer mode

Four examples are provided, two for a master and the other two for a slave with job mode disabled and enabled.

#### 16.6.3.1 Transmit/receive in master mode when job mode is disabled

The procedure below is based on the assumption that:

- The transmission data length is 8-bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CHABnCFGx.CSIHnDIRx = 0).
- Default clock and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is disabled (CSIHnCTL1.CSIHnJE = 0).
- The number of data packets is 9 (CSIHnMCTL2.CSIHnND[7:0] = $09_H$).
- The transfer start address is $10_H$ (CSIHnMCTL2.CSIHnSOP[6:0] = $10_H$).

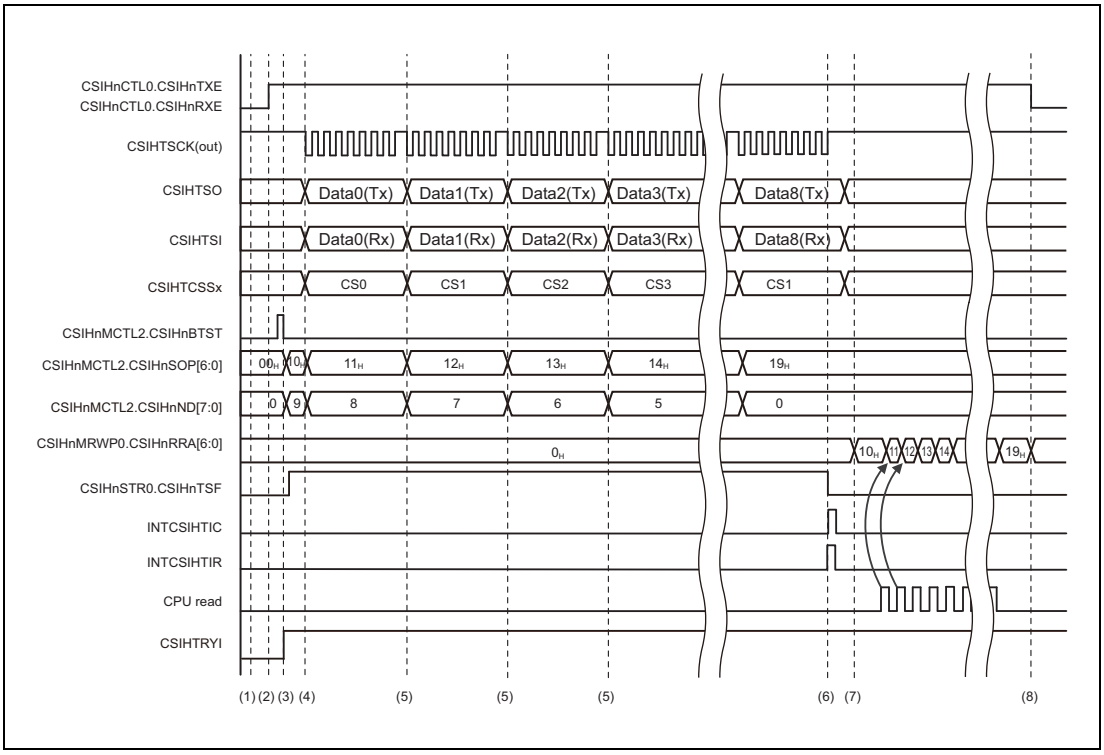



**Figure 16.55 Master in dual buffer mode, CSIHnCTL1.CSIHnJE = 0**

**NOTE**

The process of writing the data into the buffer is not described.

Set CSIHnMRWP0.CSIHnTRWA6-0 to the first data address and write the transmit data to CSIHnTX0W. The CSIHnMRWP0.CSIHnTRWA6-0 is automatically incremented after each write.

**Procedure:**

1. Before setting CSIHnCTL0.CSIHnPWR = 1, configure the required bits of following registers at first. CSIHnCTL1, CSIHnCTL2, CSIHnMCTL0, CSIHnCFG0 to 7.
2. Set bits of CSIHnCTL0.CSIHnPWR = 1 (enable the clock), CSIHnCTL0.CSIHnTXE = 1 (enable transmission), CSIHnCTL0.CSIHnRXE = 1 (enable reception), CSIHnCTL0.CSIHnMBS = 0 (Memory mode).
3. Configure the send pointer and the number of data by setting bits CSIHnMCTL2.CSIHnSOP[6:0] and CSIHnMCTL2.CSIHnND[7:0].
   Start communication by setting CSIHnMCTL2.CSIHnBTST.
4. Communication is started. Bits CSIHnMCTL2.CSIHnSOP[6:0] are automatically incremented and bits CSIHnMCTL2.CSIHnND[7:0] decremented after each data.
5. This is repeated until the last data is transmitted/received. The INTCSIHTIC and INTCSIHTIR are not generated.
6. When all communications are completed, the INTCSIHTIC and INTCSIHTIR are generated. The CPU starts to read the received data from the Rx buffer.
7. The start address of the read access is specified in CSIHnMRWP0.CSIHnRRA[6:0] (CSIHnRRA[6:0] is set to $10_H$ by the software in this figuer). These bits are incremented after the reading of every data.
8. To finalize, disable the transmit/receive operation, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE. In order to minimize the power consumption of CSIH while it is not used, set also CSIHnCTL0.CSIHnPWR = 0.

### 16.6.3.2 Transmit/receive in master mode when job mode is enabled

The procedure below is based on the assumption that:

- The transmission data length is 8-bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is enabled (CSIHnCTL1.CSIHnJE = 1).
- The number of data packets is 8 (CSIHnMCTL2.CSIHnND[7:0] = $08_H$).
- The transfer start address is $00_H$ (CSIHnMCTL2.CSIHnSOP[6:0] = $00_H$).

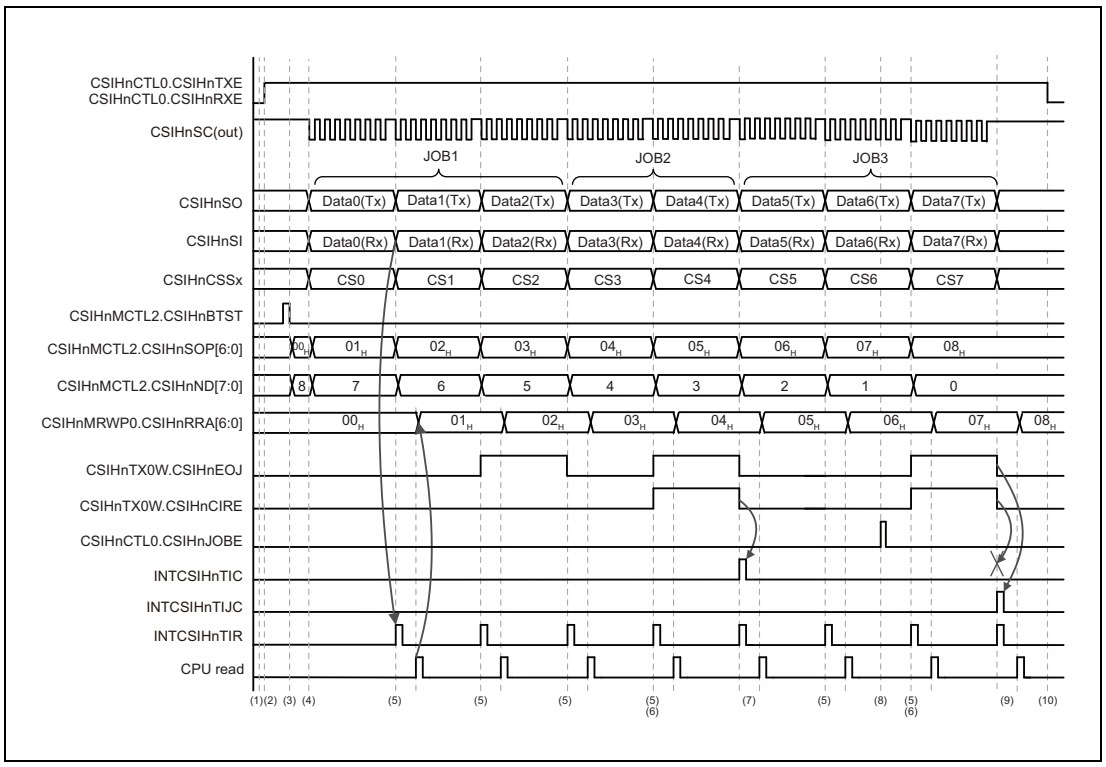



**Figure 16.56 Master in dual buffer mode, CSIHnCTL1.CSIHnJE = 1**

**NOTES**

1. The process of writing the data into the buffer is not described.
   Set CSIHnMRWP0.CSIHnTRWA6-0 to the first data address and write the transmit data to CSIHnTX0W. The CSIHnMRWP0.CSIHnTRWA6-0 is automatically incremented after each write.
2. The int_JOBE signal in the timing chart above is an internal signal for the CSIHnJOBE bit.

**Procedure:**

1. Before setting CSIHnCTL0.CSIHnPWR = 1, configure the required bits of following registers at first. CSIHnCTL1, CSIHnCTL2, CSIHnMCTL0, CSIHnCFG0 to 7.
2. 2. Set bits of CSIHnCTL0.CSIHnPWR = 1 (enable the clock), CSIH0CTL0.CSIH0TXE = 1 (enable transmission), CSIHnCTL0.CSIHnRXE = 1 (enable reception), CSIHnCTL0.CSIHnMBS = 0 (Memory mode).
3. Configure the send pointer and the number of data by setting bits CSIHnMCTL2.CSIHnSOP[6:0] and CSIHnMCTL2.CSIHnND[7:0].
   Start communication by setting CSIHnMCTL2.CSIHnBTST.
4. Communication is started. Bits CSIHnMCTL2.CSIHnSOP[6:0] are automatically incremented and bits CSIHnMCTL2.CSIHnND[7:0] decremented after each data.
5. After every data has been received, the INTCSIHTIR is generated. INTCSIHTIR indicates that the reception register CSIHnRX0W must be read.
6. The INTCSIHTIC is generated. INTCSIHTIC indicates that the last data of the current job (CSIHnTX0W.CSIHnEOJ = 1) was sent with CSIHnTX0W.CSIHnCIRE = 1.
7. By setting CSIHnCTL0.CSIHnJOBE = 1 the communication is forced to stop at the end of JOB3.
8. After the forced stop of communication, the INTCSIHTIJC and INTCSIHTIR are generated at the end of JOB3. The INTCSIHTIJC indicates a forced stop of communication at the end of the current job. The INTCSIHTIC is not generated because the INTCSIHTIJC is generated instead. Additionally, the transmission data available in register CSIHnTX0W is not sent.
9. To finalize, disable the transmit/receive operation, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE. In order to minimize the power consumption of CSIH while it is not used, set also CSIHnCTL0.CSIHnPWR = 0.

### 16.6.3.3 Transmit/receive in slave mode when job mode is disabled

The procedure below is based on the assumption that:

- The transmission data length is 8-bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock and data phase (CSIHnCFG0.CSIHnCKP0 = 0, CSIHnCFG0.CSIHnDAP0 = 0)
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is disabled (CSIHnCTL1.CSIHnJE = 0).
- The number of data packets is 9 (CSIHnMCTL2.CSIHnND[7:0] = $09_H$).
- The transfer start address is $10_H$ (CSIHnMCTL2.CSIHnSOP[6:0] = $10_H$).

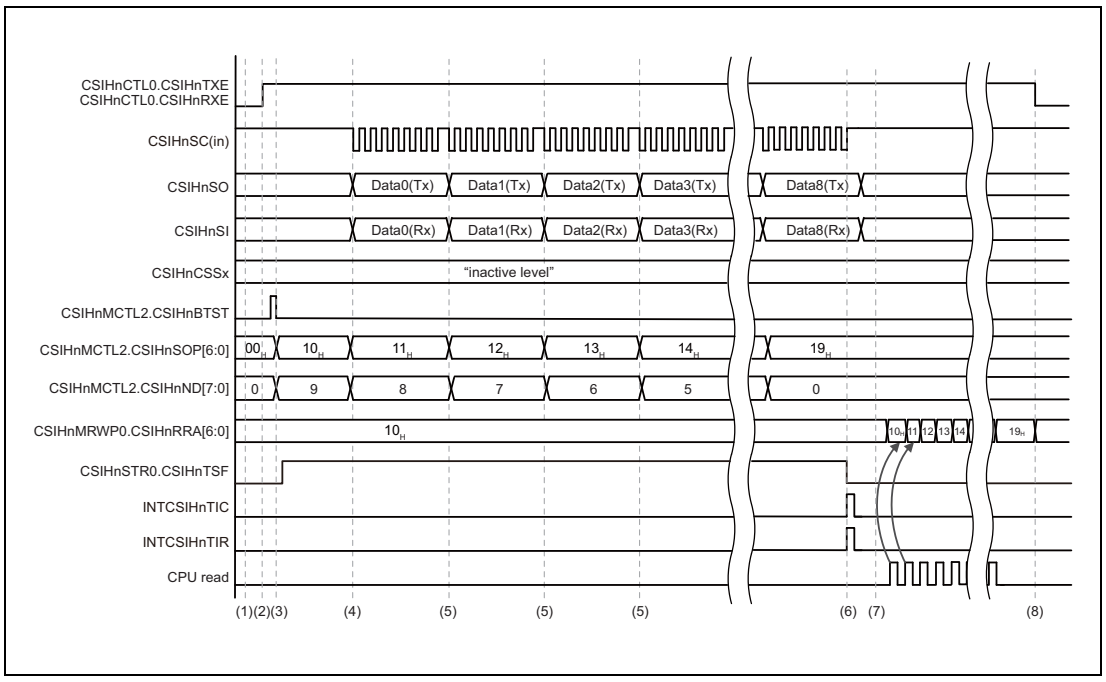



**Figure 16.57 Slave in dual buffer mode, CSIHnCTL1.CSIHnJE = 0**

**NOTE**

The process of writing the data into the buffer is not described.
Set CSIHnMRWP0.CSIHnTRWA6-0 to the first data address and write the transmit data to CSIHnTX0W. The CSIHnMRWP0.CSIHnTRWA6-0 is automatically incremented after each write.

**Procedure:**

1. Before setting CSIHnCTL0.CSIHnPWR = 1, configure the required bits of following registers at first. CSIHnCTL1, CSIHnCTL2, CSIHnMCTL0, CSIHnCFG0 to 7.
2. Set bits of CSIHnCTL0.CSIHnPWR = 1 (enable the clock), CSIHnCTL0.CSIHnTXE = 1 (enable transmission), CSIHnCTL0.CSIHnRXE = 1 (enable reception), CSIHnCTL0.CSIHnMBS = 0 (Memory mode).
3. Configure the send pointer and the number of data by setting bits CSIHnMCTL2.CSIHnSOP[6:0] and CSIHnMCTL2.CSIHnND[7:0].
   Start communication by setting CSIHnMCTL2.CSIHnBTST.
4. Communication is started when serial clock comes from master. Bits CSIHnMCTL2.CSIHnSOP[6:0] are automatically incremented and bits CSIHnMCTL2.CSIHnND[7:0] decremented after each data.
5. This is repeated until the last data is transmitted/received. The INTCSIHTIC and INTCSIHTIR are not generated.
6. When all communications are completed, the INTCSIHTIC and INTCSIHTIR are generated. The CPU starts to read the received data from the Rx buffer.
7. The CPU starts to read the received data from the Rx buffer (CSIHnRRA[6:0] is set to $10_H$ by the software in this figure). The start address of the read access is specified in CSIHnMRWP0.CSIHnRRA[6:0]. These bits are incremented after the reading of every data.
8. To finalize, disable the transmit/receive operation, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE. In order to minimize the power consumption of CSIH while it is not used, set also CSIHnCTL0.CSIHnPWR = 0.

### 16.6.4 Procedures in FIFO mode

Two examples for a master is provided, one with job mode disabled, the other one with job mode enabled.

#### 16.6.4.1 Transmit/receive in master mode when job mode is disabled

The procedure below is based on the assumption that:

- The transmission data length is 8-bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0).
- Normal clock and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is disabled (CSIHnCTL1.CSIHnJE = 0).

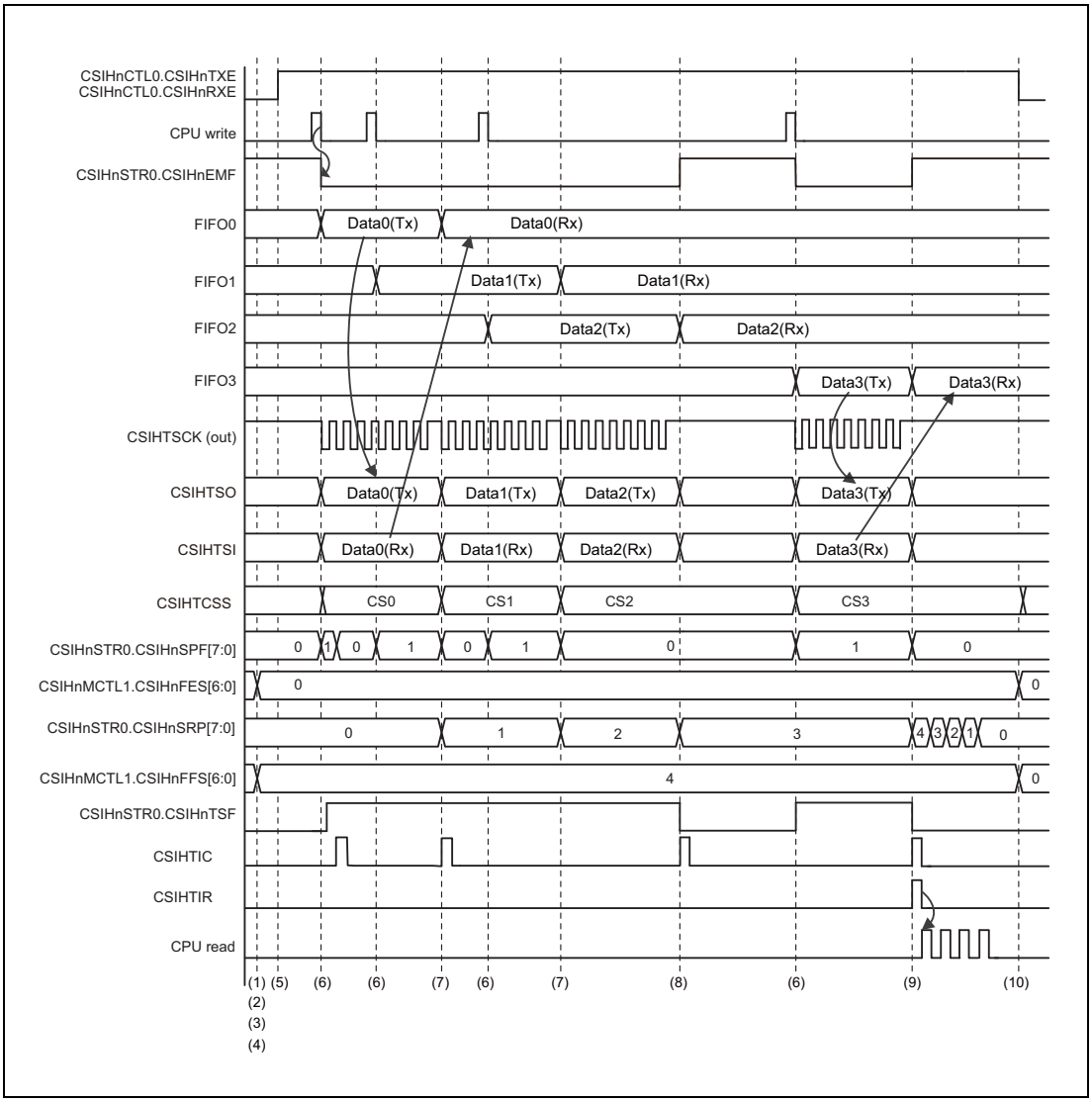



**Figure 16.58 Master in FIFO mode, CSIHnCTL1.CSIHnJE = 0**

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. This example uses chip select signals CS0 to CS3.
2. Specify the transfer and operating modes by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
3. Set CSIHnMCTL0.CSIHnMMS[1:0] = $00_B$ (FIFO mode).
   Set bit CSIHnSTCR0.CSIHnPCT to 1 to clear all buffer pointers.
   Make sure CSIHnSTR0.CSIHnFLF is set to 0, CSIHnSTR0.CSIHnEMF is set to 1, and CSIHnSTR0.CSIHnSPF[7:0] is set to $00_H$.
   With CSIHnMCTL1.CSIHnFES[6:0], specify the conditions for the CSIHTIC interrupt and the FIFO empty flag (CSIHnSTR0.CSIHnEMF[6:0]).
   With CSIHnFFS[6:0] in the same register, specify the condition for the CSIHTIR interrupt and the FIFO full flag (CSIHnSTR0.CSIHnFLF).
4. In the CSIHnCTL0 register, set bits CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), and CSIHnRXE = 1 (permits reception). The CSIHnCTL0.CSIHnMBS bit must be cleared.
5. Write the first data to be sent to the transmission register, CSIHnTX0. Transmission starts automatically when the first data becomes available.
   Make sure CSIHnSTR0.CSIHnEMF is set to 0.
6. The current transmission is complete.
7. Bit CSIHnSTCR0.CSIHnEMF is set to 1 when there is no further data to be transferred into the buffer.
   The interrupt request, CSIHTIC is generated because CSIHnFES[6:0] = CSIHnSPF[7:0].
8. The CSIHTIR is generated because values of CSIHnMCTL1.CSIHFFS[6:0] and (128 - CSIHnSTR0.CSIHnSRP[7:0]) are equal at this timing. The CSIHTIC is generated because values of CSIHnSTR0.CSIHnSPF[7:0] and CSIHnMCTL1.CSIHnFES[6:0] are equal at this timing.
   The CPU starts to read the received data that are stored in the buffer.
9. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation.

### 16.6.4.2 Master in transmit/receive mode when job mode is enabled

The procedure below is based on the assumption that:

- The transmission data length is 8-bits (CSIHnCFGx.CSIHnDLSx[3:0] = $1000_B$).
- The transmission direction is MSB first (CSIHnCFGx.CSIHnDIRx = 0)
- Normal clock and data phase (CSIHnCFGx.CSIHnCKPx = 0, CSIHnCFGx.CSIHnDAPx = 0)
- No general interrupt delay (CSIHnCTL1.CSIHnSIT = 0)
- Job mode is enabled (CSIHnCTL1.CSIHnJE = 1).
- JOB1 consists of four data packets, JOB2 consists of three data packets, and JOB3 consists of five data packets.

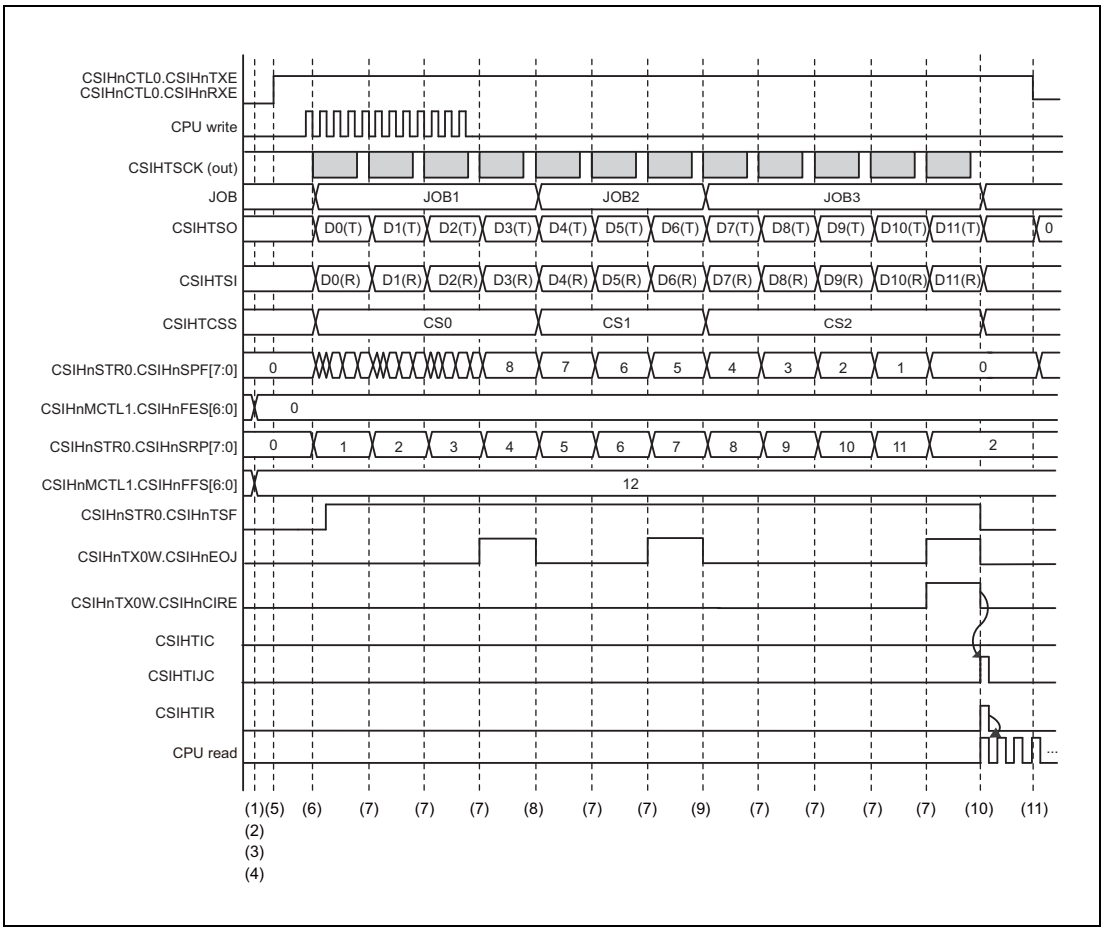



**Figure 16.59 Master in FIFO mode, CSIHnCTL1.CSIHnJE = 1**

**NOTE**

"int_JOBE" in the above figure is internal signal of CSIHnCTL0.CSIHnJOBE bit.

**Procedure:**

1. Configure the communication protocol in the CSIHnCFGx register. The example uses chip select signals CS0 to CS2.
2. Select the memory mode by setting CSIHnMCTL0.CSIHnMMS[1:0] = $00_B$.
   Specify the transfer and operating modes by setting the corresponding bits in registers CSIHnCTL1 and CSIHnCTL2.
3. Set bit CSIHnSTCR0.CSIHnPCT to 1 to clear all buffer pointers.
4. Make sure CSIHnSTR0.CSIHnFLF is set to 0, CSIHnSTR0.CSIHnEMF is set to 1, and CSIHnSTR0.CSIHnSPF[7:0] is set to $00_H$.
   With CSIHnMCTL1.CSIHnFES[6:0], specify the conditions for generating the CSIHTIC interrupt request. With CSIHnMCTL1.CSIHnFFS[6:0], specify the conditions for generating the CSIHTIR interrupt request.
5. In the CSIHnCTL0 register, set bits CSIHnPWR = 1 (enables the clock), CSIHnTXE = 1 (permits transmission), and CSIHnRXE = 1 (permits reception). The CSIHnCTL0.CSIHnMBS bit must be cleared.
6. Write the first data packet to be sent to the CSIHnTX0 transmission register. Transmission starts automatically when the first data becomes available.
   Make sure CSIHnSTR0.CSIHnEMF is set to 0.
7. The current transmission is completed.
8. Setting CSIHnTX0W.CSIHnEOJ = 1 specifies that the last data of the current job is sent.
   The CSIHTIC interrupt request is not generated because the last data of the current job (CSIHnTX0W.CSIHnEOJ = 1) was sent with CSIHnTX0W.CSIHnCIRE = 0.
9. Setting CSIHnCTL0.CSIHnJOBE to 1 forcibly stops communication when JOB3 ends.
10. After communication was stopped forcibly, interrupt requests CSIHTIJC and CSIHTIR are generated at the end of JOB3.
    The CSIHTIJC interrupt request indicates that communication is forcibly terminated when the current job ends.
    Interrupt request CSIHTIC is not generated because interrupt request CSIHTIJC is generated instead. Additionally, the transmission data available in the CSIHnTX0H register is not sent.
11. Finally, clear CSIHnCTL0.CSIHnTXE and CSIHnCTL0.CSIHnRXE to prohibit the transmit/receive operation.

## 16.7 Priority Management Module (PMM)

CSIH Priority Management Module (PMM) is an add-on module of CSIH module with transfer groups & priority handling capability. The PMM will handle up to 8 TG's with six configurable priorities. Each TG will be able to generate two DTS trigger which can be used to load transmit data from local or global RAM and store receive data into local or global RAM. (DTS configuration must be done in software). PMM will enable software to use the same CSIH by different applications simultaneously, without any software synchronization.

### 16.7.1 Transfer groups

The CSIH communication will be handled by transfer groups (TG).

Each TG has the following properties:

- Configured with one or more jobs
- Six priority levels can be set
- DTS triggers dedicated for receive path and transmit path are used.
  - Can be changed even when another transfer group (TG) is under transferring
  - Can be changed without affecting another transfer group (TG)
  - Can be halted between jobs by the transfer group (TG) with a higher priority
  - Three types of triggers
    Hardware trigger
    Software trigger
    Hardware and software trigger

### 16.7.1.1 TG structure

The general message format within a TG is shown in **Figure 16.60, TG message format**.

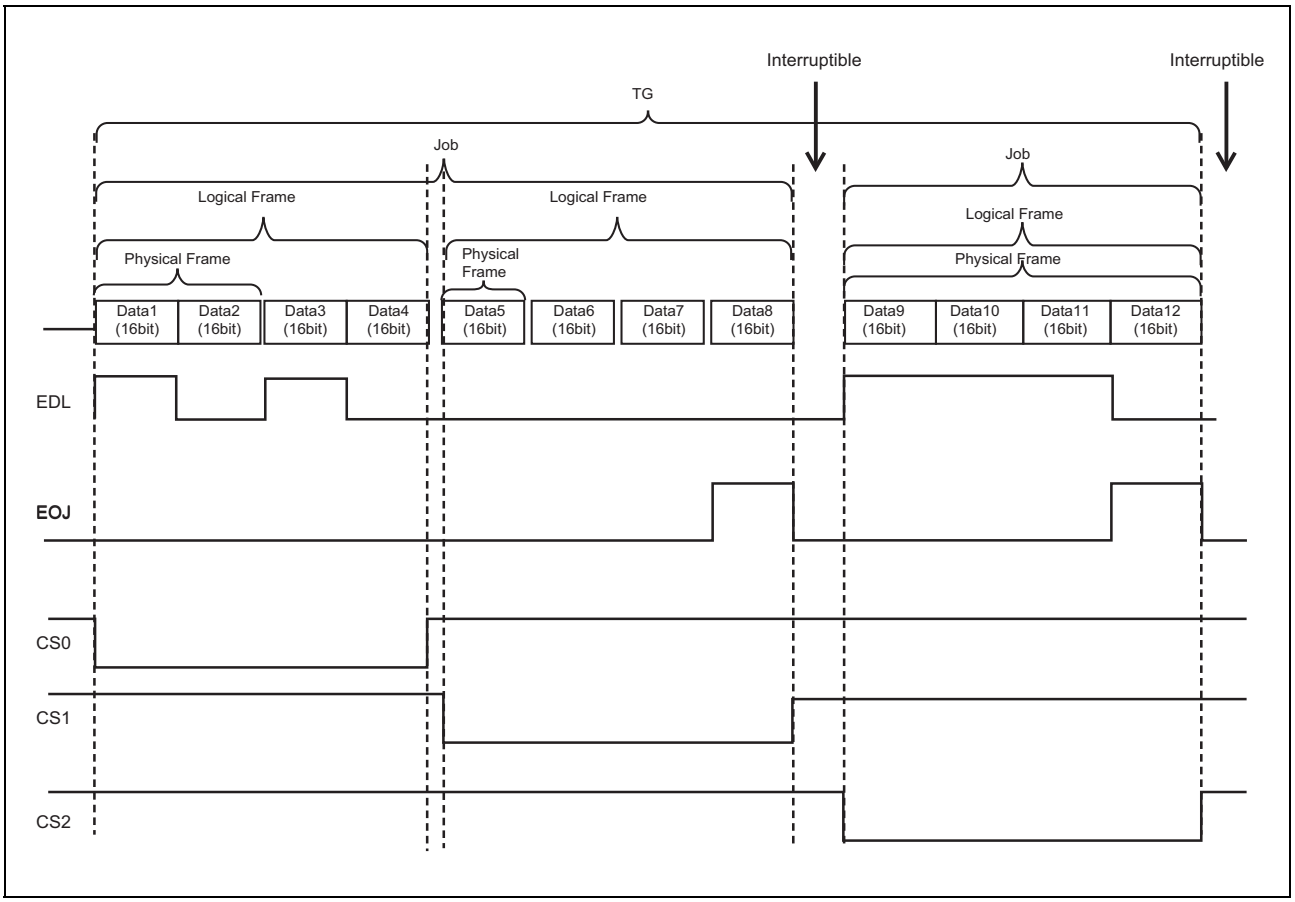



**Figure 16.60 TG message format**

The TG can be separate in the following units (beginning with the smallest unit):

- Physical frame
- Logical frame
- Job

### (1) Physical frame

A physical frame is the smallest atomic transfer of data. The length of the physical frame is flexible (2 bits up to n bit). A physical frame with more than 16 bits will be transferred by means of the EDL feature of the CSIH. For details, see the CSIH description. Within one physical frame the “CS setting” (active CS signal, transfer direction, parity usage, clock/data phase,… see CSIH documentation for details) will not be changed.

The physical frame cannot be interrupted by a higher priority transfer.

### (2) Logical frame

A logical frame is a composition of multiple physical frames. The “CS settings” may change between the included physical frames. The logical frame cannot be interrupted by a higher prior transfer.

### (3) Job

A Job is a composition of multiple logical frames. The end of a Job must be indicated by setting the EOJ bit in the upper 16 bit of the last Tx data.

After each job the communication is interruptible by a higher priority communication request.

### 16.7.1.2 TG status

The following different statuses are used to indicate the current status of a TG. See **Section 16.7.4, PMM operation** details.

- **Disabled [PMMAnENm = 0; PMMAnPFm = 0; PMMAnAFm = 0; PMMAnTFm = X; PMMAnRFm = X]**
  TG is not used and is not considered for priority handling and data transfer.
  In this mode the configuration settings in PMMAnTCTLm register can be modified.
  This mode will be entered after a reset occurs and it is the final state after TG transfer.
- **In case of Tx/Rx mode:** The PMMAnTFm and PMMAnRFm flag will be set.
  This indicates that all Tx data was sent to CSIH and all Rx data was read by DTS. TG can be reconfigured and enabled for next transfer.
  See **Section 16.7.5.2** for details.
- **In case of Tx only mode:** only the PMMAnTFm will be set.
  This indicates that all Tx data was sent to CSIH.

**NOTE**

In case of transition to Disabled mode by set of PMMAnSPm (SW abortion) PMMAnTFm and PMMAnRFm will be set to 0 and PMMTIJC will be generated.

In case of Tx/Rx mode user must wait until data was transferred to the RAM, before reconfiguration for next transfer.

Transition to disabled mode is possible from each mode, except Active mode and wait mode, by setting the STOP trigger bit (PMMAnSPm) in TG control register (PMMAnTCTLm).

If the STOP bit is set for a TG which is currently in the Active mode the TG will be set to disable mode after finishing the current Job.

All needed preconfigurations in RAM, DTS and PMM are done by software and PMM waits for initial trigger.

- **Enabled [PMMAnENm = 1; PMMAnPFm = 0; PMMAnAFm = 0; PMMAnTFm = 0; PMMAnRFm = 0]**
  All needed preconfigurations in RAM, DTS and PMM are done by SW and PMM waits for initial trigger.
- **Pending [PMMAnENm = 1; PMMAnPFm = 1; PMMAnAFm = 0; PMMAnTFm = 0; PMMAnRFm = 0]**
  TG was triggered by SW or HW and Job transfer is pending.
- **Active [PMMAnENm = 1; PMMAnPFm = 1; PMMAnAFm = 1; PMMAnTFm = 0; PMMAnRFm = 0]**
  A Job of TG is currently processed in CSIH and PMM.

**NOTE**

Only one TG can be active at the same time.

- **Wait [PMMAnENm = 1; PMMAnPFm = 0; PMMAnAFm = 0/1; PMMAnTFm = X; PMMAnRFm = 0]**
  In case of transition to wait mode during normal communication, the PMMAnTFm will be 1.
  In case of transition due to set of PMMAnSPm (SW abortion) PMMAnTFm will be 0.

**In case of Tx/Rx mode:**

Data transmission of last TG Job to the CSIH has completed.
As long as the last Rx data was not received in PMMAnRX PMMAnAFm = 1.
When Rx data was received in PMMAnRX, but data was not read by DTS, PMMAnAFm will change to 0.

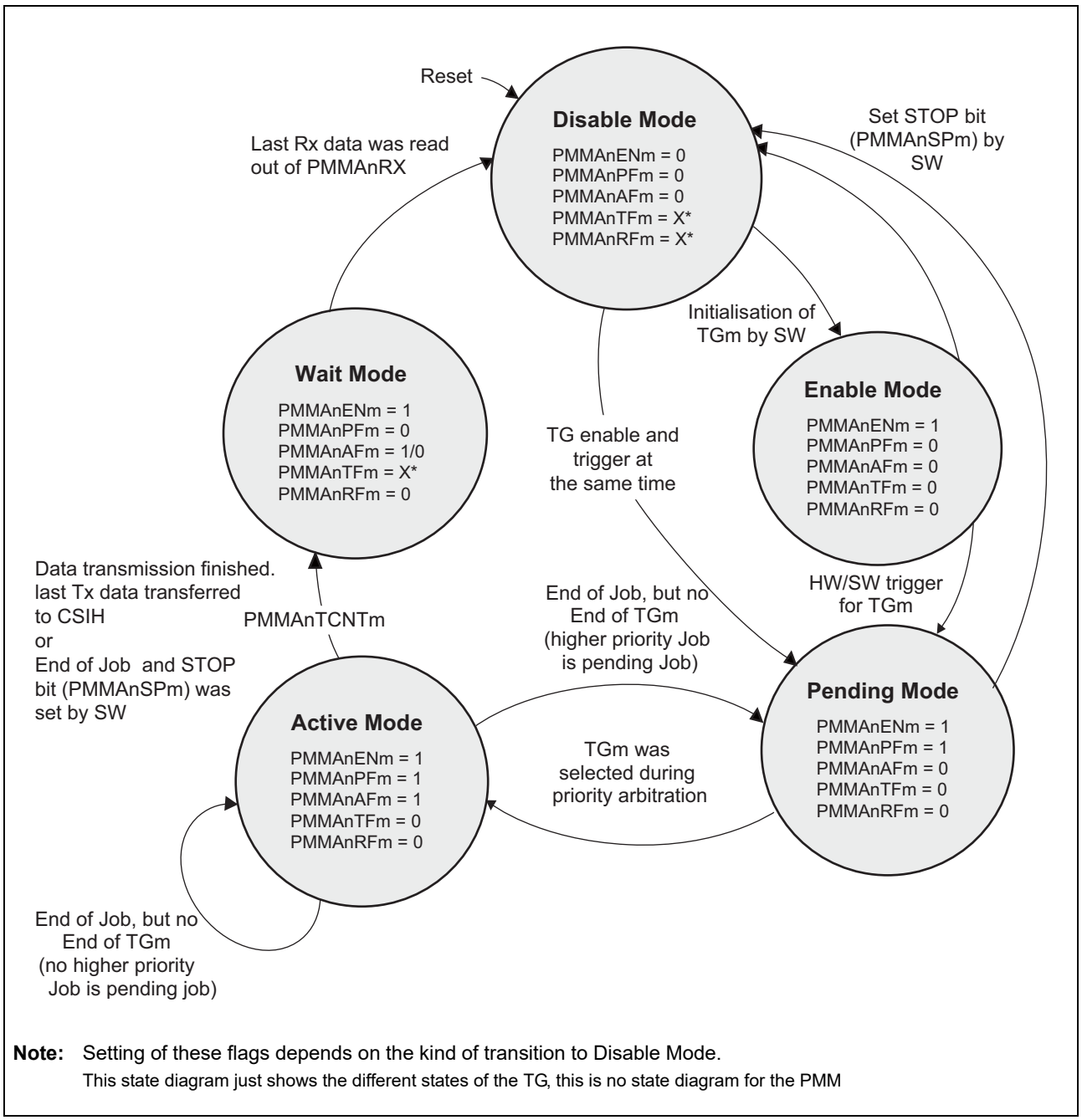
**In case of Tx only mode:**

Data transmission of last TG Job to the CSIH has completed.
As long as the last “dummy” Rx data was not received in PMMAnRX PMMAnAFm = 1.
When Rx data was received in PMMAnRX, PMMAnAFm will change to 0 and TG will enter disable mode.

**NOTE**

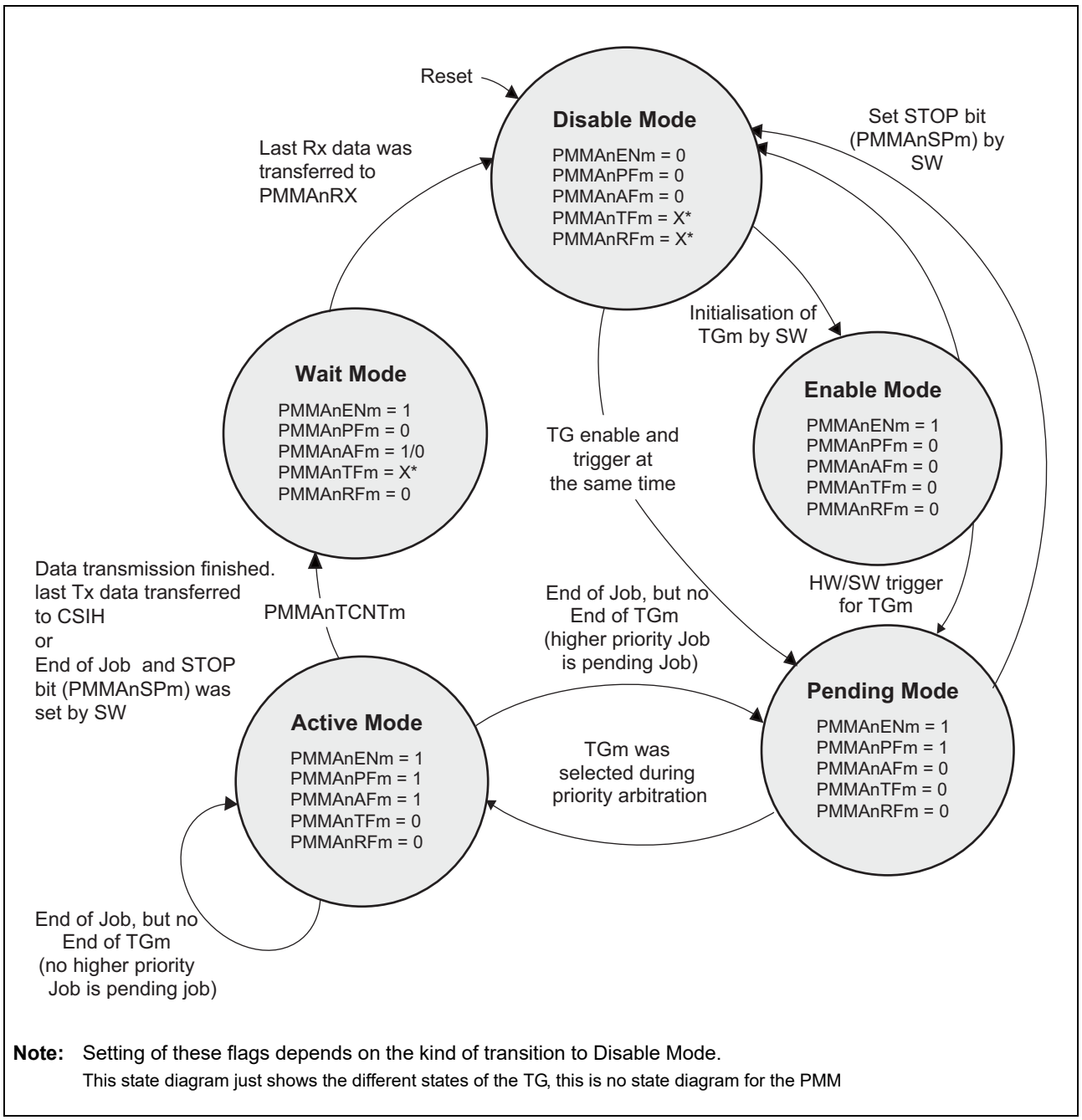
To improve the data throughput the next pending TG will be started when PMMAnAFm changes from 1 to 0.

“dummy” Rx data will be send to PMMAnRX, because CSIH operates always in Tx/Rx mode.




**Note:** Setting of these flags depends on the kind of transition to Disable Mode.
This state diagram just shows the different states of the TG, this is no state diagram for the PMM

**Figure 16.61 TG state diagram (Tx/Rx mode)**




**Note:** Setting of these flags depends on the kind of transition to Disable Mode.
This state diagram just shows the different states of the TG, this is no state diagram for the PMM

**Figure 16.62 TG state diagram (Tx only mode)**

### 16.7.1.3 TG Flexible Priority

Priority assignment of each TG's will be done by setting PMMAnPRIOm[2:0] bits of PMMAnTCTLm register. Setting a value of 0 gives the TG the highest priority and setting a value of 5 gives the TG the lowest priority. If two or more TG has same PMMAnPRIOm[2:0] bit setting, then the natural priority will be considered. That is, TG0 has the highest priority and TG7 has the lowest priority.

Writing to PMMAnPRIOm[2:0] bits of PMMAnTCTLm register is only allowed when the TGm is in the disable mode.

### 16.7.2 Block Diagram

The block diagram shows the main components of the PMM.

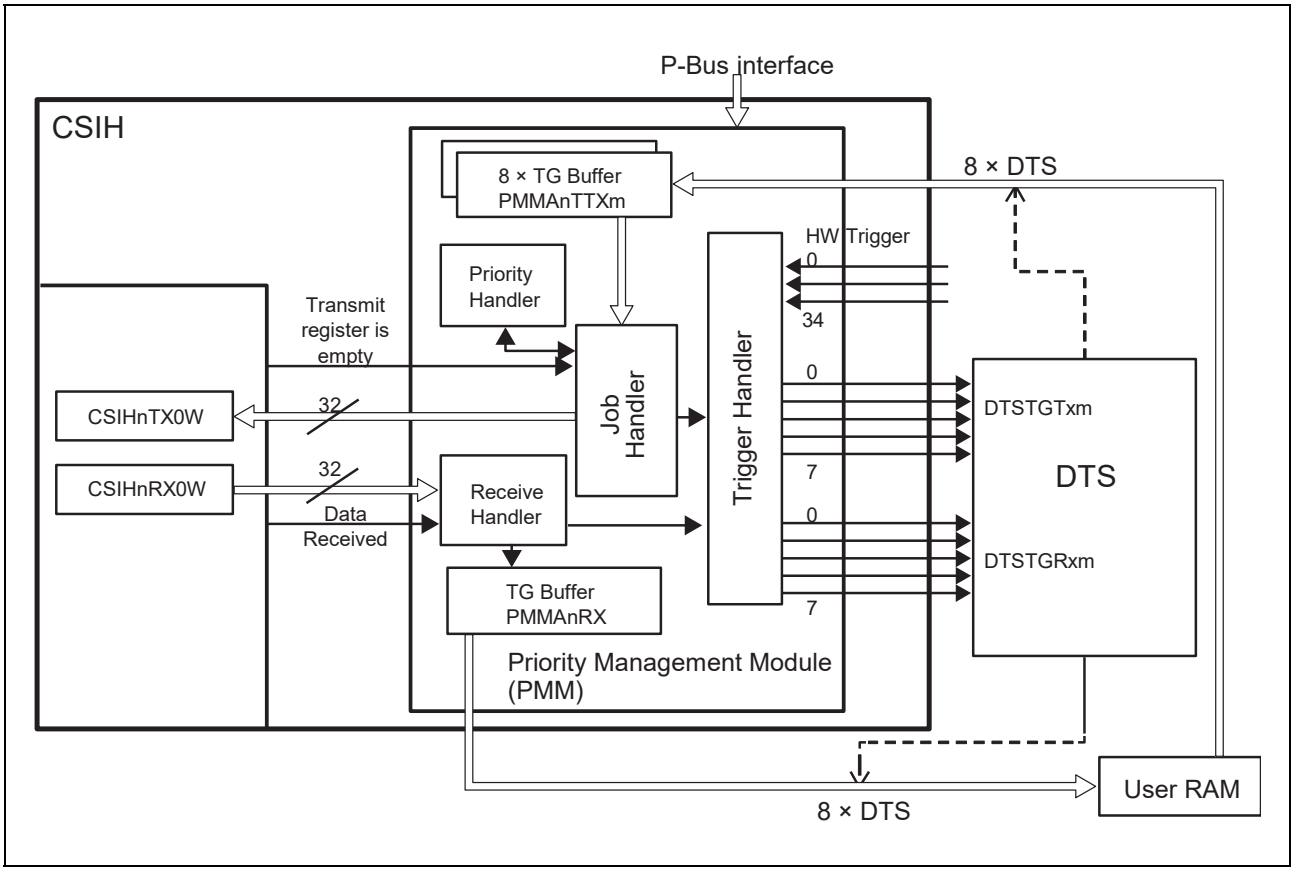



Figure 16.63 Block diagram of PMM

### 16.7.3 Functional Overview

- One PMM per CSIH instance
- Support of up to 8 transfer groups, each with six configurable priority levels
- Three interrupt request signals
  - PMMTIC: Data transfer to CSIH of TG finished
  - PMMRIC: Data reception in CSIH and PMM finished (Tx/Rx mode). Data transfer finished (Tx only mode).
  - PMMTIJC: TG was aborted by PMMAnSPm bit.

Each TG has the following capability

- Generation of DTS trigger signals for receive and transmit path
- Software and hardware triggered start of TG transfer
- Selectable hardware trigger source from multiple input signals
- Enable/Disable TG
- Support TG length of up to 256 × 32bit (data + control bits)
- Support of EDL functionality of CSIH
- Configurable job length by EOJ bit in control bits of the Tx data

**NOTE**

PMMTIC is shared with CSIHTIC

PMMRIC is shared with CSIHTIR

PMMTIJC is shared with CSIHTIJC

#### 16.7.3.1 Function description

The PMM can manage the communication of up to 8 different TG's. The user only has to set up the TG structure in RAM and configure the DTS and PMM accordingly. As soon as the TG was triggered by hardware or software, the PMM handles the complete communication between local RAM and CSIH for transmit and receive communication. Data transfer between local RAM and PMM is done by DTS and peripheral bus. Therefore the PMM can trigger up to 16 different DTS channels, 8 for transmit and 8 for receive path. The transfer from PMM to CSIH is done directly, without bus usage.

The start of a TG communication can be triggered by software and/or hardware. The PMM has up to 35 hardware trigger inputs which can be assigned to the different TG's. (Each TG can be assigned to only one hardware or software trigger)

As soon as a TG was triggered, the PMM will process the first Job of the TG. To handle also more than one pending TG, the PMM process priority arbitration after each Job end to serve always the highest prioritized TG next.

#### 16.7.3.2 TG buffer

To minimise delay time, due to peripheral bus load, and to handle DTS lock phase the PMM is equipped with 9 data buffers, one transmit buffer (PMMAnTTXm) for each TG and one common receive buffer (PMMAnRX). In PMMAnTTXm the data from local RAM to CSIH are buffered. In PMMAnRX the received data from CSIH are buffered before sending them to local RAM via DTS. Each buffer contains 32bits to store the complete CSIHTx0W and CSIHRx0W content.

**NOTE**

To avoid any loss of data, due to a delay in reading the received data from PMMAnRX.

The PMM is equipped with a flow control, which stops loading data to CSIH in case of receive path is fully loaded.

#### 16.7.3.3 Trigger Handler

The DTS trigger logic handles the incoming hardware trigger signals, the software trigger and it generates the DTS trigger signals for each TG.

**(1) Trigger inputs**

The user application can trigger the communication of a TG by following trigger options

- Software trigger
- Hardware trigger
- Hardware trigger and software trigger

Following external hardware trigger sources will be supported. The used hardware trigger can be set in PMMAnTCTLm register separately for each TG.

**NOTE**

One Trigger can be assigned to several TG. In this case each TG will be triggered by this trigger. If a hardware or software trigger will occur for an already started TG the trigger will be ignored by the PMM.

**Table 16.53 PMM External trigger sources (1/2)**

| PMMAnHWTm[5:0] | Unit | Trigger source |
|---|---|---|
| 000000 | Pin | External Interrupt 0 (INTP0) |
| 000001 | Pin | External Interrupt 1 (INTP1) |
| 000010 | Pin | External Interrupt 2 (INTP2) |
| 000011 | Pin | External Interrupt 3 (INTP3) |
| 000100 | STM0 | STM0_interrupt 0 |
| 000101 | STM0 | STM0_interrupt 1 |
| 000110 | GTM | Interrupt 0 from sub module MCS0 |
| 000111 | GTM | Interrupt 1 from sub module MCS0 |
| 001000 | GTM | Interrupt 2 from sub module MCS0 |
| 001001 | GTM | Interrupt 3 from sub module MCS0 |
| 001010 | GTM | Interrupt 4 from sub module MCS0 |
| 001011 | GTM | Interrupt 5 from sub module MCS0 |
| 001100 | GTM | Interrupt 0 from sub module MCS1 |
| 001101 | GTM | Interrupt 1 from sub module MCS1 |
| 001110 | GTM | Interrupt 0 from sub module ATOM0 |
| 001111 | GTM | Interrupt 1 from sub module ATOM0 |
| 010000 | GTM | Interrupt 2 from sub module ATOM0 |
| 010001 | GTM | Interrupt 3 from sub module ATOM0 |
| 010010 | GTM | Interrupt 4 from sub module ATOM0 |
| 010011 | GTM | Interrupt 5 from sub module ATOM0 |
| 010100 | GTM | Interrupt 6 from sub module ATOM0 |
| 010101 | GTM | Interrupt 7 from sub module ATOM0 |
| 010110 | GTM | Interrupt 0 from sub module ATOM1 |
| 010111 | GTM | Interrupt 1 from sub module ATOM1 |
| 011000 | GTM | Interrupt 2 from sub module ATOM1 |
| 011001 | GTM | Interrupt 3 from sub module ATOM1 |

**Table 16.53 PMM External trigger sources (2/2)**

| PMMAnHWTm[5:0] | Unit | Trigger source |
|---|---|---|
| 011010 | GTM | Interrupt 4 from sub module ATOM1 |
| 011011 | GTM | Interrupt 5 from sub module ATOM1 |
| 011100 | GTM | Interrupt 6 from sub module ATOM1 |
| 011101 | GTM | Interrupt 7 from sub module ATOM1 |
| 011110 | GTM | Interrupt 0 from sub module ATOM2 |
| 011111 | GTM | Interrupt 1 from sub module ATOM2 |
| 100000 | GTM | Interrupt 2 from sub module ATOM2 |
| 100001 | GTM | Interrupt 3 from sub module ATOM2 |
| 100010 | GTM | Interrupt 4 from sub module ATOM2 |
| Other than above | — | Setting prohibited |

**NOTE**

When setting all except for 000000-100010 as PMMAnHWTm[5:0], a trigger becomes invalid.

## 16.7.4 PMM operation

### 16.7.4.1 Input trigger handling

After enabling the TG (see **Section 16.7.5.1, TG Preconfiguration:**) the Trigger Handler will observe the corresponding trigger source.

(1) When a hardware or software trigger occurs for an enabled TG, the PMM will set the TG to status pending by setting the bit PMMAnPFm. This indicates that this TG must be processed.

(2) In addition the DTSTGTXm trigger will be generated to trigger the transfer of the first data from RAM into PMMAnTTXm register.

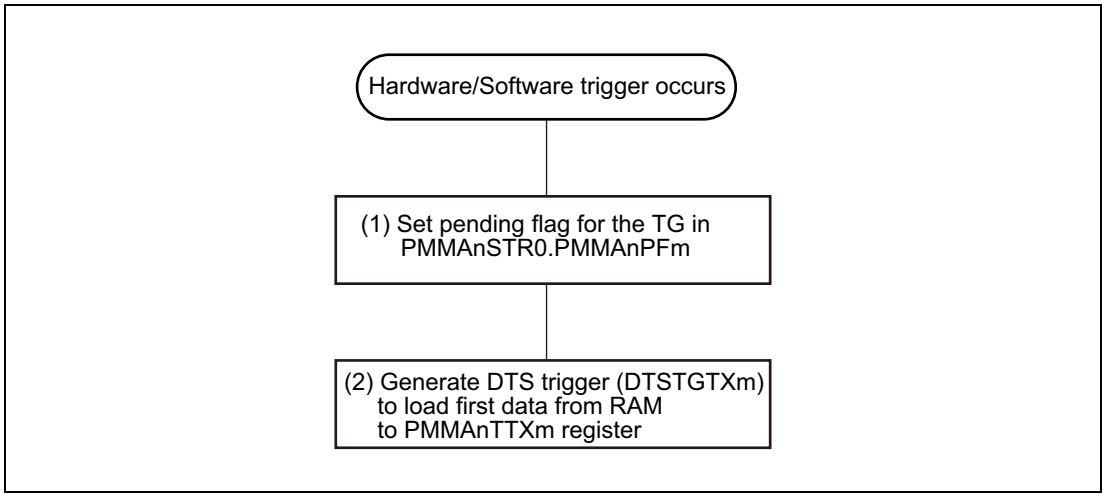



**Figure 16.64 Initial TG trigger (Input)**

**NOTE**

At the time of pending, if HW trigger and SW trigger are input, DTSTGTXm is not output.

### 16.7.4.2 Output trigger handling

The DTS trigger transmit path (DTSTGTXm) will be generated direct after initial software or hardware trigger (see **Section 16.7.4.1, Input trigger handling**) and by Job Handler (see **Section 16.7.4.4, job Handler**). Each time the Job Handler triggers the DTS the following flow will be processed. At first the Trigger Handler will check for the current active TG, then the corresponding DTS trigger will be generated.

The DTS trigger for the receive path (DTSTGRXm) will be generated only by the Receive Handler when PMM is operation in Tx/Rx mode.

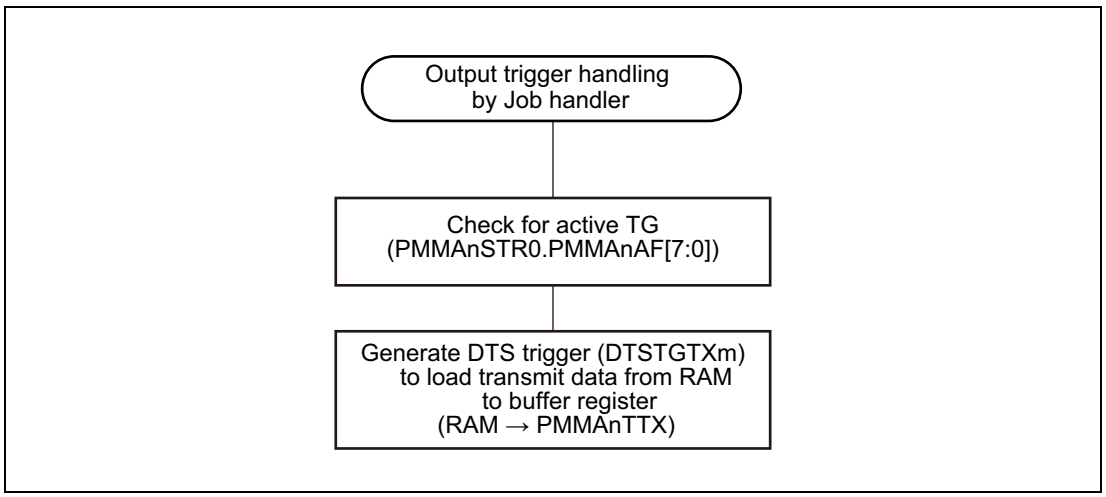



**Figure 16.65 Initial TG trigger (output)**

### 16.7.4.3 Priority Handler

The priority handler module inside the PMM detects the pending TG with the highest priority after every job end and services this TG number next.

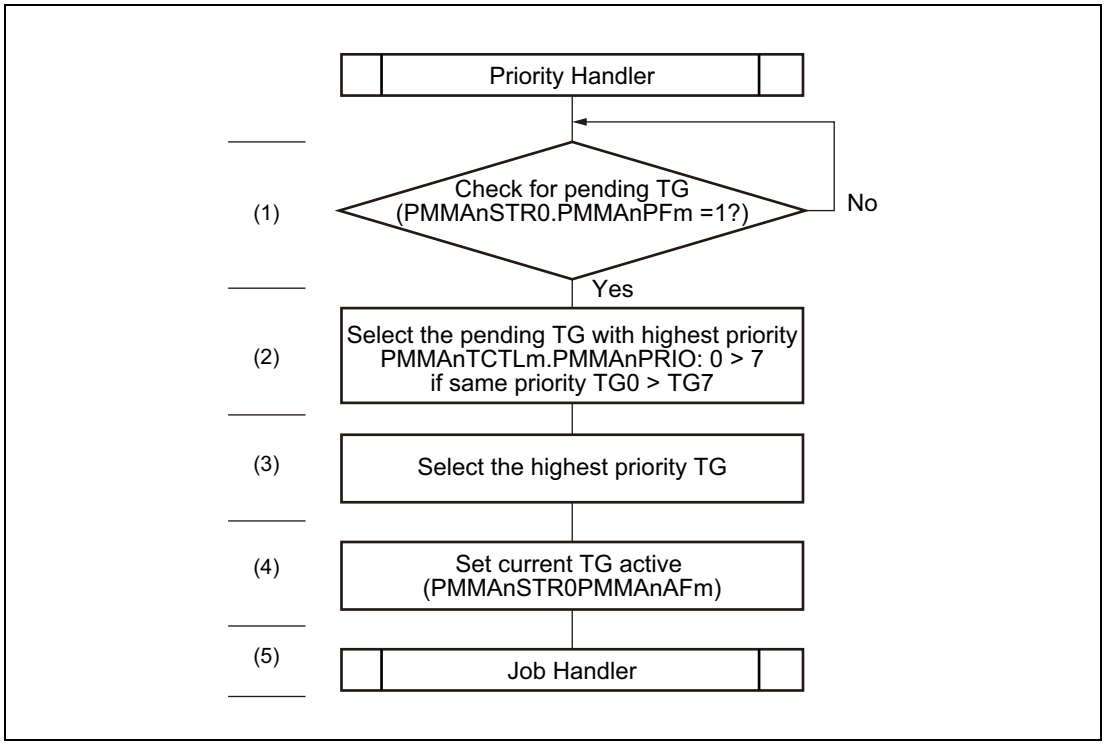



**Figure 16.66 Priority Handler flow**

(1) If no TG is pending the Priority handler will stay in idle/wait state.

(2) If there is a pending TG the Priority handler will select the pending TG with the highest set priority in the PMMAnTCTLm register, if two or more TG's have the same priority the natural priority (TG number) will be considered (TG0 has the highest priority).

(3) As the result the pending TG with the highest priority will be set active.

(4) Before starting the Job handler the selected TG will be set as active in PMMAnSTR0.PMMAnAFm.
Note: Only one TG can be active at the same time. All other PMMAnAFm bits must be cleared.

(5) Finally, the next Job of the highest prior TG will be processed in the Job handler.

Finally, the next Job of the highest prior TG will be processed in the Job handler.

After finishing the Job in the Job handler and in the Receive Handler, the Priority Handler will start from the beginning, to consider also new triggered TG's. Thereby, it is ensured that after each Job, always the next Job of the highest pending TG will be executed.

**NOTE**

When selecting the highest priority TG, even if write data isn't loaded from RAM the highest priority pending TG at the time is selected. The waiting time forms until data is loaded in TG from RAM in this case.

### 16.7.4.4 job Handler

The Job Handler is handling the transfer of data from PMM to CSIH and it will generate the DTS trigger signals via the DTS trigger module.

**NOTE**

The Job Handler will be active for the complete Job transfer. The Priority Handler will not operate during this time.

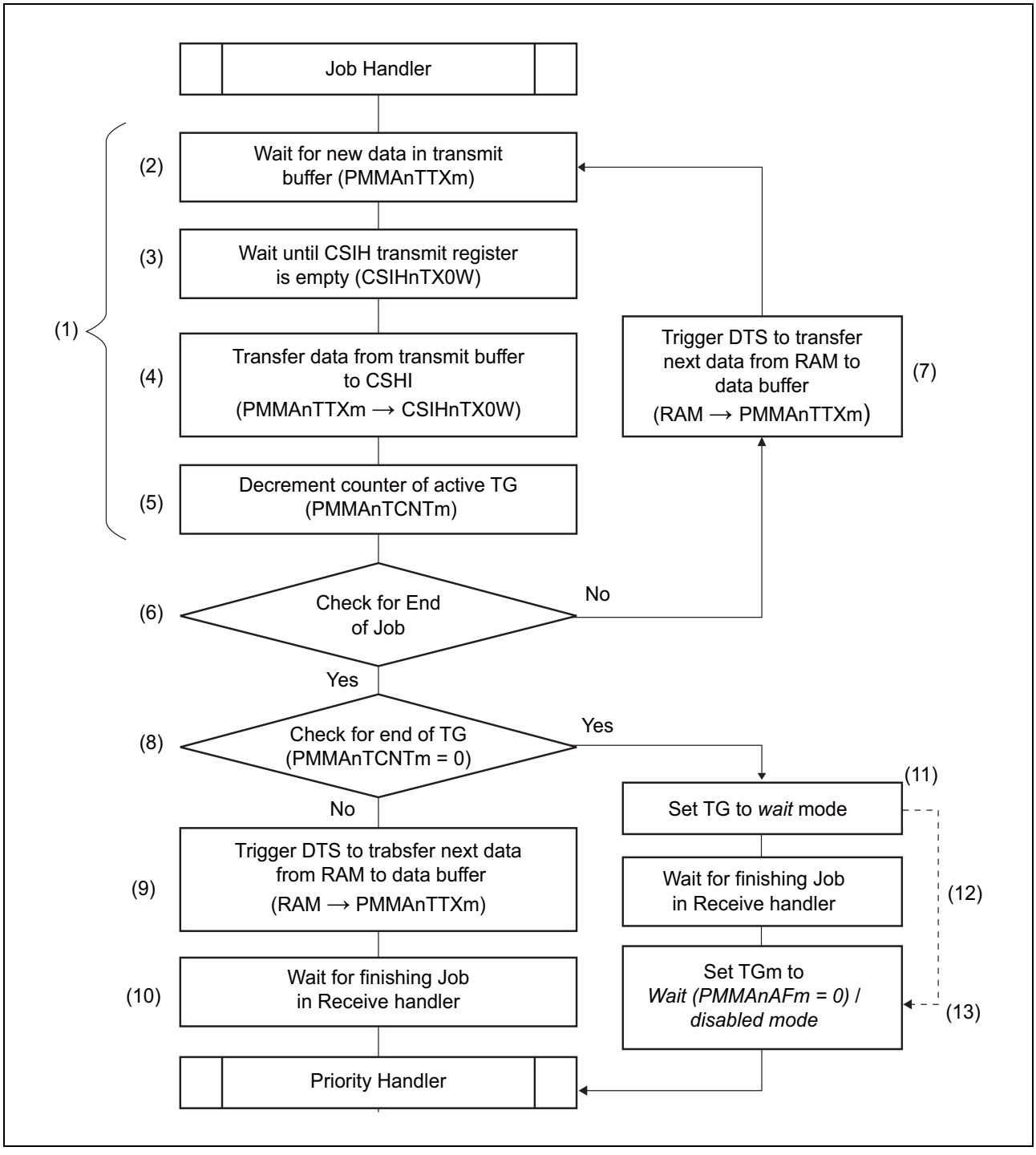



**Figure 16.67 Job Handler flow**

(1) In the main loop of the Job Handler the communication to the CSIH will be done.

(2) In each case the loop starts with a check if new data in PMMAnTTXm register are available. This check will be done to handle a possible delay between the DTS trigger and data reception in PMMAnTTXm, due to bus load or DTS lock phase.

(3) In the next step the TG Handler has to check if CSIH is ready to receive new Tx data.

(4) If both checks are done (new data are available and CSIH is ready), the Job handler will transfer the data from PMMAnTTXm to CSIH.

**NOTE**

The data of PMMAnTTXm will be treated as empty register automatically after data shift to transmit buffer register from PMMAnTTXm.

(5) After this the corresponding TG counter (PMMAnTCNTm) will be decremented to detect the end of the TG in further check.

(6) To decide if the loop (1) can be exit or not, the EOJ bit of the transferred data in step (4) will be checked.

(7) If the EOJ bit was not set, the DTSTGTXm signal of the active TG will be triggered to load the next data to PMMAnTTXm register and the loop (1) will start from the beginning.

(8) If the EOJ bit was set (Job has finished) the loop (1) will be exit and the Job Handler will check if this is also the end of the TG (TG counter = 0).

(9) If TG has not finished (TG counter ≠ 0), the DTSTGTXm trigger will be generated to pre-load the next data to PMMAnTTXm register.

(10) After this the Job Handler will wait until the last data of the active Job was processed in Receive Handler, before entering the Priority Handler.

**In case of Tx/Rx mode:**

(11) If EOJ bit was set and the TG has finished (TG counter = 0), Job Handler will set current TG to wait mode (PMMAnPFm = 0 and PMMAnTFm = 1) .

(12) The Job handler will wait until the DTS trigger for the last data of the active TG was triggered in Receive Handler.

(13) Before clearing the PMMAnAFm flag (TG still remains in wait mode).

**NOTE**

As soon as the data in PMMAnRX was read by DTS the TG will be set to disabled mode.

**In case of Tx only mode:**

(14) If EOJ bit was set and the TG has finished (TG counter = 0).

(15) Job Handler will set the PMMAnTFm in PMMAnSTR1 (9) and set the TG directly to Disabled mode.

The synchronisation (10) and (12) with the receive path has the following function:

- It ensures that always the correct DTS channel will be triggered in Receive Handler, because no other TG will be set to active before the trigger for the last DTS transfer was generated.
- It ensures that priority arbitration will not start before the Job has completely finished. Therefore, all triggers which occurred during the complete previous Job transfer will be considered in Priority Handler.
- It ensures that the PMMAnRFm flag will be set when PMM has completely processed the corresponding TG. (Including reading data from PMMAnRX register)

**NOTE**

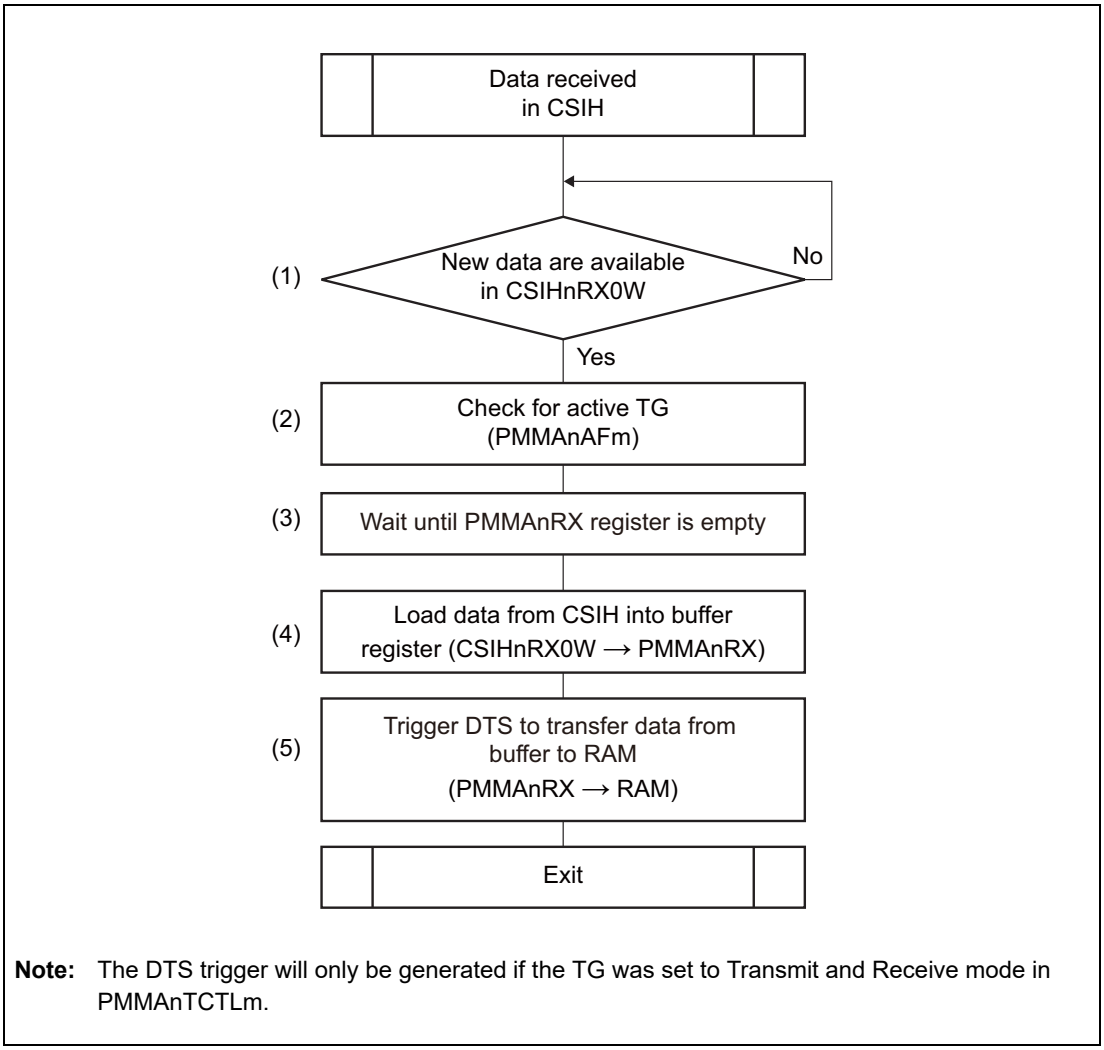
As soon as the TF flag will be set the PMMTIC interrupt will be generated.
(PMMTIC will be shared with CSIH interrupt signal)

As soon as the RF flag will be set the PMMRIC interrupt will be generated.
(PMMRIC will be shared with CSIH interrupt signal)

When the Job Handler has finished, the Priority handler starts operation from the beginning.

### 16.7.4.5 Receive Handler

When data reception in CSIH has completed and new received data are available, the Receive Handler will be automatically started.




**Note:** The DTS trigger will only be generated if the TG was set to Transmit and Receive mode in PMMAnTCTLm.

**Figure 16.68 Receive Handler flow**

(1) On the arrival of a new data in CSIHnRX0W that will be notified to PMM (1).

(2) PMM will check for the active TG.This check will avoid overwriting of data inside receive buffer.

(3) Wait until receive buffer (PMMAnRX) is empty.

(4) If the buffer is empty the Receive Handler will shift the data from CSIHnRX0W to PMMAnRX.

(5) As last action the DTS receive trigger DTSTGRXm will be generated for the active TG to load the data from PMMAnRX to local RAM and the Receive Handler will exit.

**NOTE**

The data of PMMAnRX will be treated as empty register automatically after DTS read access. Loss of Read data is stopped by making them wait for SPI communication because PMM can't stock data of CSIHnRX0W in PMMAnRX an APB bus is locked by CPU access, and when being not forwarded data to RAM from PMMAnRX.

## 16.7.5 PMM operation description

### 16.7.5.1 TG Preconfiguration:

Before using the TG and priority handling of the PMM, the following initialization must be done by application:

- Store the complete TG transmit data in RAM.
  - Including control bits for each 16 bit entry
  - Data must be stored in a consecutive order
- Set up DTS channel for transmit path
  - Source address: first transmit data of TGm in RAM (decrementing or incrementing)
  - Destination address: corresponding PMMAnTTXm register in PMM (fixed)
  - Transfer length: 32 bit
  - Count: total number of buffer entries (32 bit entry: 16 control bits + 16 data bits)
  - Trigger: DTSTGTXm from PMM
- Set up DTS channel for receive path
  - Source address: corresponding PMMAnRX register in PMM (fixed)
  - Destination address: first receive data of TGm in RAM (decrementing or incrementing)
  - Count: total number of buffer entries (32 bit entry: 16 control bits + 16 data bits)
  - Trigger: DTSTGRXm from PMM
- Configure CSIH
  - Master mode
  - Tx/Rx mode
  - Direct access mode
  - EDL in case of communication >16 bit
- Configure PMM PMMAnTCTLm
  - Select trigger option (Software and/or hardware)
  - Select hardware trigger source, in case of hardware triggering
  - Set the transfer mode (Transmit only or Transmit and Receive)

**NOTE**

If the parity check function of the CSIH will be used, it is recommended to use the PMM in Tx/Rx mode for all TG's.

Background:

CSIH will perform parity check for all TG's also for TG which will not receive data.

This will generate erroneous parity errors which will be flagged in the Rx data of the CSIH.

To detect erroneous parity errors correctly, all Rx data must be available for the user.

– Set length of TG
– Set the priority
– Clear transmit and receive status flags by PMMAnCLTFm and PMMAnCLRFm.

If all these configurations are done, the TG can be enabled in PMM PMMAnTCTLm register.

By enabling the TG the following operation will/should be executed:

- Software should ensure not to write in the control register bits (except the PMMAnSPm bit and PMMAnSWTm bit), to avoid reconfiguration of the TG during active transfer.
- PMM will load TG length in PMM counter (PMMAnTCNTm)
- PMM will clear the PMMAnTTXm register to avoid use of old data in the job handler sequence.

**NOTE**

The other TG control registers must kept unlocked to enable configuration of a new/disabled TG during PMM operation.

The data of PMMAnTTXm will be treated as empty register automatically after data shift to transmit buffer register from PMMAnTTXm.

The setting flow of Power ON/OFF of CSIH, PMM is below. Be sure to follow this flow

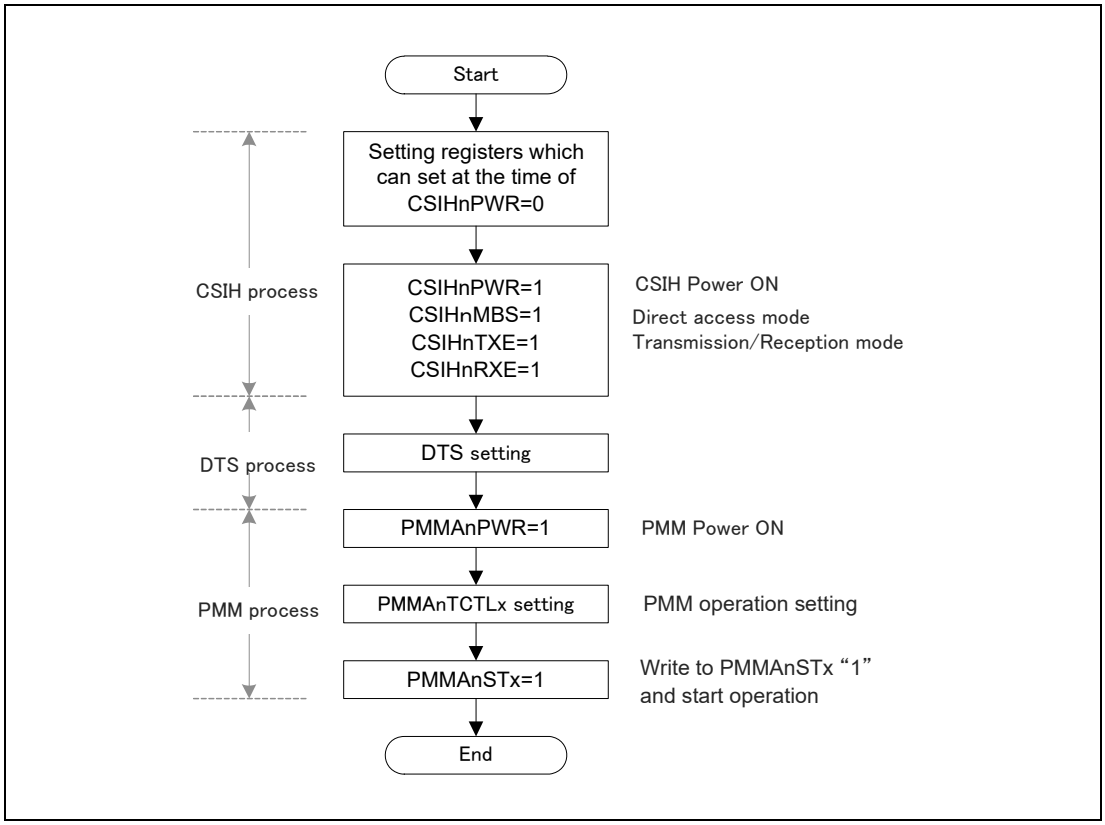



**Figure 16.69 Power on flow**

The setting of PMMAnPWR=1 and PMMAnTCTLx is inconsecutive. The setting of PMMAnTCTLx is also possible at PMMAnPWR=0.

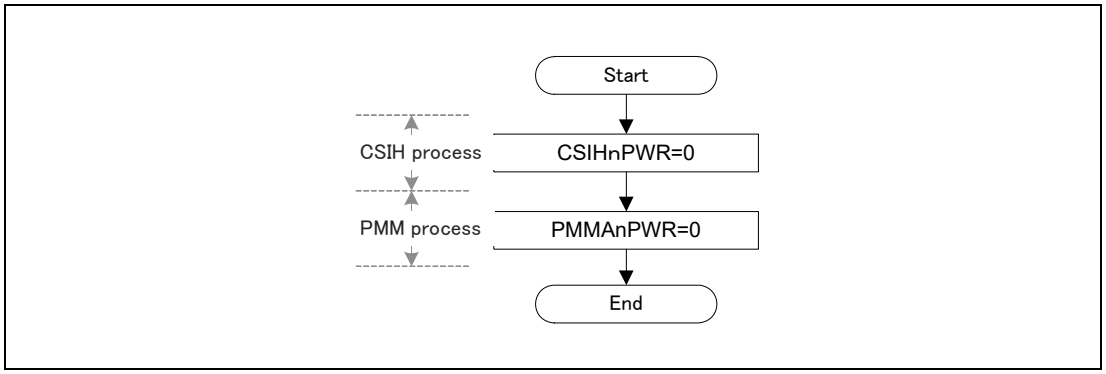



**Figure 16.70 Power off flow**

Be sure to turn off power by the order of CSIHnPWR->PMMAnPWR.

### 16.7.5.2 TG Reconfiguration:

After a TG was transferred completely and entered the disable mode, software has to do the following reconfigurations to set-up the next TG transmission:

- Set up DTS channel for transmit path
  Count: total number of buffer entries (32 bit entry: 16 control bits + 16 data bits)
- Set up DTS channel for receive path
  Count: total number of buffer entries (32 bit entry: 16 control bits + 16 data bits)
- Set up PMM
  - Select trigger option (Software and/or hardware)
  - Select hardware trigger source, in case of hardware triggering
  - Set the transfer mode (Transmit only or Transmit and Receive)
  - Set length of TG
  - Set the priority
  - Clear transmit and receive status flags by PMMAnCLTFm and PMMAnCLRFm.
  - Enable the TG by the start bit (PMMAnSTm) bit in the PMMAnTCTLm register.

## 16.8 Difference among P1M-C, P1H-C and P1H-CE

- CSIH3CSS4, CSIH3CSS5, CSIH3CSS6, CSIH3CSS7, CSIH3DCS1, CSIH3DCS2, CSIH3RYI, CSIH3RYO, CSIH3SCI1, CSIH3SCI2, CSIH3SCO1, CSIH3SCO2, CSIH3SI1, CSIH3SI2, CSIH3SO1, CSIH3SO2, CSIH3SSIZ are not available in P1M-C (QFP-144)
- CSIH3CSS4, CSIH3CSS5, CSIH3DCS2, CSIH3SCI2, CSIH3SCO2, CSIH3SI2, CSIH3SO2, CSIH3SSIZ are not available in P1M-C (BGA-292)
- CSIH0DCS1, CSIH0DCS2, CSIH0SCI1, CSIH0SCI2, CSIH0SCO1, CSIH0SCO2, CSIH0SI1, CSIH0SI2, CSIH0SO1, CSIH0SO2, CSIH1DCS2, CSIH1RYI, CSIH1RYO, CSIH1SCI2, CSIH1SCO2, CSIH1SI2, CSIH1SO2, CSIH2CSS4, CSIH2CSS5, CSIH2RYI, CSIH2RYO, CSIH3CSS1, CSIH3CSS3, CSIH3CSS4, CSIH3CSS5, CSIH3CSS6, CSIH3CSS7, CSIH3DCS1, CSIH3DCS2, CSIH3RYI, CSIH3RYO, CSIH3SCI1, CSIH3SCI2, CSIH3SCO1, CSIH3SCO2, CSIH3SI1, CSIH3SI2, CSIH3SO1, CSIH3SO2, CSIH3SSIZ are not available in P1M-C (BGA-156), P1H-C (4MB, BGA-156)
- See **Table 16.10** for difference in Data Consistency Check

# Section 17 High Speed USRT (HS-USRT)

This section describes the High-speed USRT (HS-USRT) module.

The first subsection describes all the characteristics specific to the RH850/P1x-C such as channels, register base addresses, and input/output signal names.
The second and subsequent subsections describe characteristics common to all the versions.

## 17.1 RH850/P1x-C HS-USRT Overview

### 17.1.1 Number of Channels

The RH850/P1x-C includes up to four HS-USRT channels.

**Table 17.1 HS-USRT Channels**

| | HS-USRT | |
|---|---|---|
| Product Name | P1M-C, P1H-C (4MB, BGA-156) | P1H-C (4MB, BGA-292), P1H-C (8MB), P1H-CE |
| Number of Channels | 2 | 4 |
| Name | HSUSn (n = 0 , 1) | HSUSn (n = 0 to 3) |

#### Index n

This section identifies each HS-USRT channel by “n”. For example, the HS-USRT clock enable register is described as HSUSnCKEN.

### 17.1.2 Register Addresses

HS-USRT register addresses are represented by an offset from the base address <HSUSn_base>.

The following table shows the base address <HSUSn_base> of each HS-USRT module.

**Table 17.2 Register Base Address <HSUSn_base>**

| HS-USRTn Channel | <HSUSn_base> Address |
|---|---|
| HSUS0 | FFD8 $8000_H$ |
| HSUS1 | FFCA $8000_H$ |
| HSUS2 | FFD8 $9000_H$ |
| HSUS3 | FFCA $9000_H$ |

### 17.1.3 Clock Supply

The following clock input is supplied for the HS-USRT module.

**Table 17.3 HS-USRT Clock Source**

| Module | Clock Name | Clock symbol | Description |
|---|---|---|---|
| HSUSn | High Speed system clock | CLK_HSB | High speed Bus clock (HCLK) |
| HSUSn | High Speed system clock | CLK_HSB | Peripheral Bus clock (PCLK) |
| HSUSn | High Speed Peripheral clock 2 | CLKP_H2 | Internal clock for transmission (TXCLK) |

Note 1. The max. input clock for HCKL and PCLK is 80MHz.
Note 2. The following condition must be fulfilled:
- CLK_HSB >= CLKP_H2
- CLK_HSB > SCKIO (only in case of data reception)

For detail of clock supply, see **Section 12, Clock Controller**.

### 17.1.4 Interrupts and DMA/DTS

The HS-USRT module can generate the following interrupt requests.

**Table 17.4 Interrupt Requests**

| Unit Interrupt Name | Outline | Interrupt Number | DMA Trigger Number | DTS Trigger Number |
|---|---|---|---|---|
| **HS-USRT0** | | | | |
| INTHSUS0ERR | Bundled error interrupt for HS-USRT0 | 116 | — | — |
| INTHSUS0RE | Bundled reception interrupt for HS-USRT0 | 117 | — | — |
| INTHSUS0TR | Bundled transmission interrupt for HSUSRT0 | 118 | — | — |
| **HS-USRT1** | | | | |
| INTHSUS1ERR | Bundled error interrupt for HS-USRT1 | 119 | — | — |
| INTHSUS1RE | Bundled reception interrupt for HS-USRT1 | 120 | — | — |
| INTHSUS1TR | Bundled transmission interrupt for HS-USRT1 | 121 | — | — |
| **HS-USRT2** | | | | |
| INTHSUS2ERR | Bundled error interrupt for HS-USRT2 | 122 | — | — |
| INTHSUS2RE | Bundled reception interrupt for HS-USRT2 | 123 | — | — |
| INTHSUS2TR | Bundled transmission interrupt for HS-USRT2 | 124 | — | — |
| **HS-USRT3** | | | | |
| INTHSUS3ERR | Bundled error interrupt for HS-USRT3 | 125 | — | — |
| INTHSUS3RE | Bundled reception interrupt for HS-USRT3 | 126 | — | — |
| INTHSUS3TR | Bundled transmission interrupt for HS-USRT3 | 127 | — | — |

### 17.1.5 HS-USRT Hardware Reset

The HS-USRT module and its registers are initialized by the following reset condition.

**Table 17.5 HS-USRT Reset Condition**

| Reset Name | Reset condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| All Registers | √ | √ | √ | √ | √ |

### 17.1.6 Input/Output Signals

The following table shows the HS-USRT input/output signals.

**Table 17.6 HS-USRTn Input/Output Signals (1/2)**

| Pin name | Function |
|---|---|
| **HS-USRT0** | |
| HSURT0SDIO0 | HSURT0 data I/O 0 |
| HSURT0SDIO1 | HSURT0 data I/O 1 |
| HSURT0SDIO2 | HSURT0 data I/O 2 |
| HSURT0SDIO3 | HSURT0 data I/O 3 |
| HSURT0SCKIO | HSURT0 serial clock I/O |
| HSURT0SDIR | HSURT0 direction indication output |
| HSURT0CSIO | HSURT0 chip select IO |
| **HS-USRT1** | |
| HSURT1SDIO0 | HSURT1 data I/O 0 |
| HSURT1SDIO1 | HSURT1 data I/O 1 |
| HSURT1SDIO2 | HSURT1 data I/O 2 |
| HSURT1SDIO3 | HSURT1 data I/O 3 |
| HSURT1SCKIO | HSURT1 serial clock I/O |
| HSURT1SDIR | HSURT1 direction indication output |
| HSURT1CSIO | HSURT1 chip select IO |
| **HS-USRT2** | |
| HSURT2SDIO0 | HSURT2 data I/O 0 |
| HSURT2SDIO1 | HSURT2 data I/O 1 |
| HSURT2SDIO2 | HSURT2 data I/O 2 |
| HSURT2SDIO3 | HSURT2 data I/O 3 |
| HSURT2SCKIO | HSURT2 serial clock I/O |
| HSURT2SDIR | HSURT2 direction indication output |
| HSURT2CSIO | HSURT2 chip select IO |
| HS-USRT3 | |
| HSURT3SDIO0 | HSURT3 data I/O 0 |
| HSURT3SDIO1 | HSURT3 data I/O 1 |
| HSURT3SDIO2 | HSURT3 data I/O 2 |
| HSURT3SDIO3 | HSURT3 data I/O 3 |

**Table 17.6 HS-USRTn Input/Output Signals (2/2)**

| Pin name | Function |
|---|---|
| HSURT3SCKIO | HSURT3 serial clock I/O |
| HSURT3SDIR | HSURT3 direction indication output |
| HSURT3CSIO | HSURT3 chip select IO |

## 17.2 Overview

Table 17.7 Specifications of HS-USRT

| Item | Description |
|---|---|
| Communications mode | • Half-duplex clock-synchronous serial transfer<br>• Master (transmission) or slave (reception) selectable (the SDIR pin indicates the master or slave state)<br>• MSB first<br>• Lengths of frames for transfer<br>16-bit data without parity<br>15-bit data + even parity bit<br>15-bit data + odd parity bit |
| Number of lines | 1, 2 and 4 lines selectable |
| Transfer rates | When TXCLK is running at 80 MHz:<br>• Transmission: 10 Mbps/line (40 Mbps when 4 lines are in use)<br>Reception: 40 Mbps/line (160 Mbps when 4 lines are in use) |
| Master communications clock division settings | Ratios for frequency division of TXCLK are selectable from among 1/8, 1/10, 1/12, 1/14, 1/16, 1/24, and 1/32. |
| Transfer buffer capacities | • Transmission: 8 frames (2 × 4-frame buffers)<br>• Reception: 64 frames (2 × 32-frame buffers) |
| • Interrupt sources | • Packet transmission complete interrupt<br>• First frame reception interrupt<br>• Group frame reception interrupt<br>• Packet reception complete interrupt<br>• Packet error interrupt<br>• Receive overflow error interrupt |
| Error detection | • Parity error (none, odd, even)<br>• Receive overflow error |
| Other functions | • Internal DMAC to handle transfer of data in communications<br>• Register write protection |
| Clock signals | • Internal clock for transmission: 60/80 MHz<br>• Peripheral bus clock : 60/80 MHz<br>• High-speed bus clock : 40/80 MHz<br>• Rx clock (SCKIO): 40 MHz |

### 17.2.1 Block Diagram

The following figure shows a block diagram of the HS-USRT module.

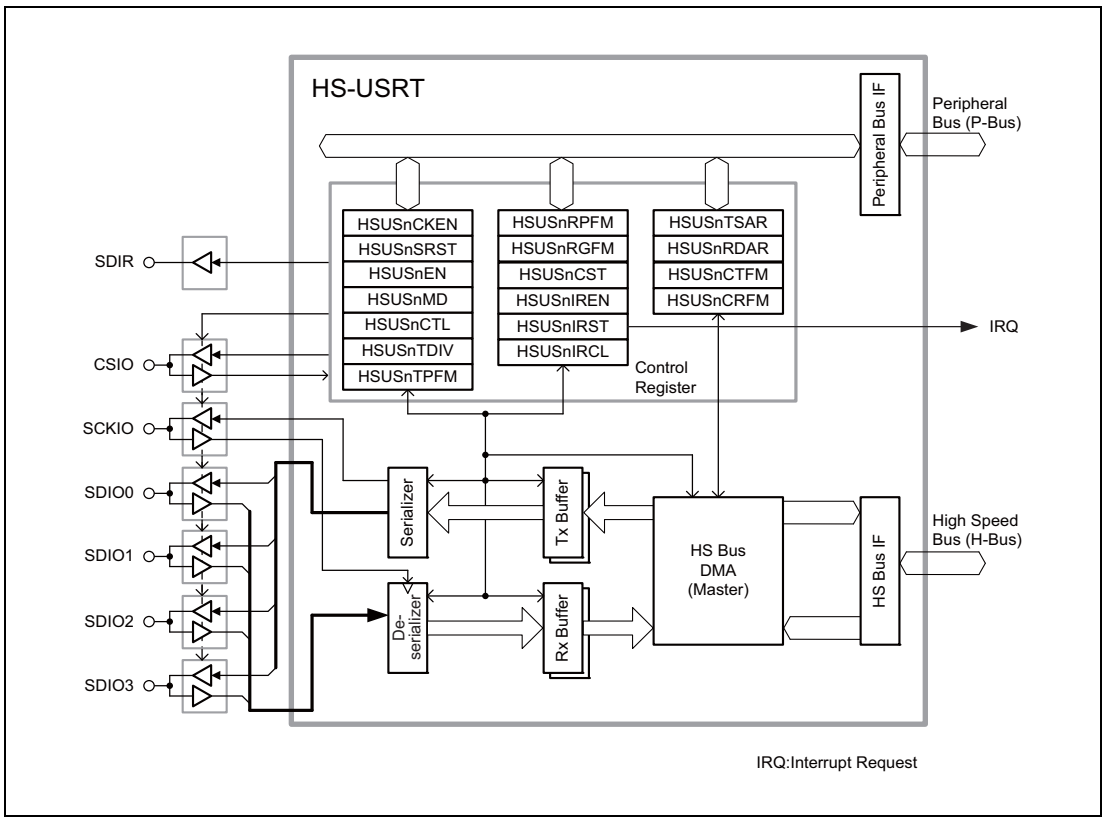



**Figure 17.1 Block Diagram of HS-USRT**

The LSI pin name have prefix for identification (see **Section 17.1.6, Input/Output Signals**).

## 17.3 Registers

HS-USRTn is controlled and operated by the following registers.

**Table 17.8 Overview of HS-USRTn Registers**

| Register Name | Abbreviation | Address | Access Protection | |
|---|---|---|---|---|
| | | | PBG | Others |
| HS-USRT Clock Enable Register | HSUSnCKEN | <HSUSn_base> + $0000_H$ | *1 | — |
| HS-USRT Software Reset Register | HSUSnSRST | <HSUSn_base> + $0004_H$ | *1 | — |
| HS-USRT Communication Enable Register | HSUSnEN | <HSUSn_base> + $0008_H$ | *1 | — |
| HS-USRT Mode Register | HSUSnMD | <HSUSn_base> + $000C_H$ | *1 | — |
| HS-USRT Communication Control Register | HSUSnCTL | <HSUSn_base> + $0010_H$ | *1 | — |
| HS-USRT Transmit Clock Division Select Register | HSUSnTDIV | <HSUSn_base> + $0014_H$ | *1 | — |
| HS-USRT Transmit Data Source Address Register | HSUSnTSAR | <HSUSn_base> + $0018_H$ | *1 | — |
| HS-USRT Transmit Frame Number Register | HSUSnTPFM | <HSUSn_base> + $001C_H$ | *1 | — |
| HS-USRT Data Destination Address Register | HSUSnRDAR | <HSUSn_base> + $0020_H$ | *1 | — |
| HS-USRT Receive Frame number Register | HSUSnRPFM | <HSUSn_base> + $0024_H$ | *1 | — |
| HS-USRT Receive Group Frame Number Register | HSUSnRGFM | <HSUSn_base> + $0028_H$ | *1 | — |
| HS-USRT Interrupt Enable Register | HSUSnIREN | <HSUSn_base> + $002C_H$ | *1 | — |
| HS-USRT Interrupt Status Register | HSUSnIRST | <HSUSn_base> + $0030_H$ | *1 | — |
| HS-USRT Interrupt Status Clear Register | HSUSnIRCL | <HSUSn_base> + $0034_H$ | *1 | — |
| HS-USRT Communication Status Register | HSUSnCST | <HSUSn_base> + $0038_H$ | *1 | — |
| HS-USRT Current Transmit Frame Number Register | HSUSnCTFM | <HSUSn_base> + $003C_H$ | *1 | — |
| HS-USRT Current Receive Frame Number Register | HSUSnCRFM | <HSUSn_base> + $0040_H$ | *1 | — |

Note 1. In the case of
n = 0 PBG3#1.PG3-HSUSRT0,
n = 1 PBG1#1.PG1-HSUSRT1,
n = 2 PBG3#1.PG3-HSUSRT2,
n = 3 PBG1#1.PG1-HSUSRT3

### <HSUSn_base>

The base address <HSUSn_base> of HS-USRTn is defined by the keywords in **Table 17.2, Register Base Address <HSUSn_base>**.

### 17.3.1 HSUSnCKEN — HS-USRT Clock Enable Register

This register is used to control HS-USRT clock operations.

**Access:** This register can be read/written in 32-bit units.

**Address:** <HSUSn_base> + $0000_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HSUSn CKEN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 17.9 HSUSnCKEN register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | HSUSnCKEN | Clock Operation Enable Bit<br>Control the operation of HCLK and of TXCLK.<br>0: Stop the clock supply (Except peripheral clock and SCKIO).<br>1: Enable the clock operation. |

The value of this register can be changed while communication is halted (HSUSnEN.HSUSnEN bits = 0 and HSUSnCST.HSUSnACTF bits = 0).

#### HSUSnCKEN Bit (Clock Operation Enable)

This bit runs and stops clock signals.

The clock signals to be controlled are HCLK and TXCLK.

### 17.3.2 HSUSnSRST — HS-USRT Software Reset Register

This register is used to initiate a software reset.

**Access:** This register can be written only in 32-bit units.

**Address:** <HSUSn_base> + $0004_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HSUSn SRST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 17.10 HSUSnSRST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | HSUSnSRST | Software Reset Bit<br>All registers other than the HSUSnSRST register are initialized by writing '1' to the bit. This bit always reads as 0.<br>0: No function<br>1: Cause reset of HS-USRT by Software. |

A software reset can be initiated while communication is halted (HSUSnEN.HSUSnEN bit = 0 and HSUSnCST.HSUSnACTF bit = 0).

**NOTE**

After software reset generation, Read this register at once before any register access.

#### HSUSnSRST Bit (Software Reset)

When a software reset is initiated, all registers of HS-USRT except this register are initialized.

Reading this bit always returns 0.

### 17.3.3 HSUSnEN — HS-USRT Communication Enable Register

This register enables or disables communication.

**Access:** This register can be read/written in 32-bit units.

**Address:** <HSUSn_base> + $0008_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HSUSnEN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 17.11 HSUSnEN register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | HSUSnEN | Communication Enable Bit<br>Enable or disable the communication.<br>0: Disable the communication<br>1: Enable the communication. |

#### HSUSnEN Bit (Communication Enable)

This bit enables or disables communication.

In combination with a setting in the communication mode register (HSUSnMD), this bit controls the state of the SDIR pin for specifying the direction of transfer and of the bidirectional (I/O) pins that handle transfer. For details on the states of the SDIR pin and bidirectional pins, see **Table 17.27, Settings for Mode and Enable Bits and Pin Control**.

If 0 is written to this bit during communication, ongoing communication is aborted. If 0 is written during communication, follow the procedure in **Section 17.4.9.5, Aborting Transfer**.

Write access to the protected registers is not allowed while communication is enabled.

### 17.3.4 HSUSnMD — HS-USRT Mode Register

This register selects the communication direction (transmission or reception).

**Access:** This register can be read/written in 32-bit units.

**Address:** <HSUSn_base> + $000C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HSUSnMD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 17.12 HSUSnMD register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | HSUSnMD | Communication Mode Bit<br>Control the operation direction of the HS-USRT.<br>0: Transmission<br>1: Reception |

The value of this register can be changed while communication is halted (HSUSnEN.HSUSnEN bits = 0 and HSUSnCST.HSUSnACTF bits = 0).

#### HSUSnMD Bit (Communication Mode)

This bit controls the direction of transfer (transmission or reception) by the HS-USRT.

In combination with a setting in the communication enable register (HSUSnEN), this bit controls the state of the SDIR pin for specifying the direction of transfer and of the bidirectional (I/O) pins that handle transfer. For details on the states of the SDIR pin and bidirectional pins, see **Table 17.27, Settings for Mode and Enable Bits and Pin Control.**

### 17.3.5 HSUSnCTL — HS-USRT Communication Control Register

This register controls the handling of parity and selects the number of lines (SDIO lines) used in transfer.

**Access:** This register can be read/written in 32-bit units.

**Address:** <HSUSn_base> + $0010_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | PRSL[1:0] | | — | — | LANE[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R/W | R/W |

**Table 17.13 HSUSnCTL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5, 4 | HSUSn PRSL[1:0] | Parity Select Bit<br>Select the usage of parity (no parity/even parity/odd parity) for the communication data.<br>$00_B$: No parity (transmission) / Parity is not checked (reception)<br>$01_B$: Even parity<br>$10_B$: Setting prohibited<br>$11_B$: Odd parity |
| 3, 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | HSUSn LANE[1:0] | Number of SDIO Lines Select Bit<br>Select the number of SDIO lanes for communication.<br>$00_B$: SDIO0, SDIO1, SDIO2, SDIO3<br>$01_B$: SDIO0, SDIO1<br>$10_B$: SDIO0<br>$11_B$: Setting prohibited |

The value of this register can be changed while communication is halted (HSUSnEN.HSUSnEN bits = 0 and HSUSnCST.HSUSnACTF bits = 0).

#### HSUSnLANE[1:0] Bits (number of SDIO Lines Select)

This bit selects the number of lines used in transfer.

For the lines to be used and data arrangement in memory, see **Section 17.4.3.1, Relationship between the Format of Data Storage and the Lines**.

#### HSUSnPRSL[1:0] Bits (Parity Select)

This bit selects the handling of parity (no parity, even parity, or odd parity) for data in transfer.

### 17.3.6 HSUSnTDIV — HS-USRT Transmit Clock Division Select Register

This register selects the frequency division ratio for transmission.

**Access:** This register can be read/written in 32-bit units.

**Address:** <HSUSn_base> + $0014_H$

**Value after reset:** $0000\ 0003_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | HSUSnTDIV[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 17.14 HSUSnTDIV register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | HSUSn TDIV[3:0] | Transmission Clock Division Selection Bit<br>Select the division factor for the transmission operation.<br>$0011_B$: TXCLK / 8<br>$0100_B$: TXCLK / 10<br>$0101_B$: TXCLK / 12<br>$0110_B$: TXCLK / 14<br>$0111_B$: TXCLK / 16<br>$1011_B$: TXCLK / 24<br>$1111_B$: TXCLK / 32<br>Other than above: Setting prohibited |

The value of this register can be changed while communication is halted (HSUSnEN.HSUSnEN bits = 0 and HSUSnCST.HSUSnACTF bits = 0).

#### HSUSnTDIV[3:0] Bits (Transmission Clock Division Select)

This bit selects the division factor for transmission.

### 17.3.7 HSUSnTSAR — HS-USRT Transmit Data Source Address Register

This register sets the source address of data for transmission.

**Access:** This register can be read/written in 32-bit units.

**Address:** <HSUSn_base> + $0018_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSUSnTSAR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSUSnTSAR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.15 HSUSnTSAR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | HSUSn TSAR[31:0] | Transmit data Source Address Bit<br>Define the read pointer base address of the transmit data packet.<br>Source Address of the transmit data packet. |

The value of this register can be changed while communication is halted (HSUSnEN.HSUSnEN bits = 0 and HSUSnCST.HSUSnACTF bits = 0).

#### HSUSnTSAR[31:0] Bit (Transmit Data Source Address)

These bits set the source address of data for transmission.

Since data from the memory is read in 4-frame units, the write value to the 3 lower-order bits is ignored, and reading these bits always returns 0.

## 17.3.8 HSUSnTPFM — HS-USRT Transmit Frame Number Register

This register specifies the number of frames for transmission.

**Access:** This register can be read/written in 32-bit units.

**Address:** <HSUSn_base> + $001C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | HSUSnTPFM[12:0] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.16 HSUSnTPFM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 12 to 0 | HSUSn TPFM[12:0] | Transmit Frame Number Bits<br>Specify the number of transmit frames.<br>Up to 4 K frames can be specified ($0001_H$ to $1000_H$).<br>Setting the value beyond above range is not allowed.<br>4 lines are used: The multiple number of 4 can be specified.<br>2 lines are used: The multiple number of 2 can be specified.<br>1 line is used: Any value can be specified. |

The value of this register can be changed while communication is halted (HSUSnEN.HSUSnEN bits = 0 and HSUSnCST.HSUSnACTF bits = 0).

### HSUSnTPFM[12:0] Bits (Transmit Frame number)

These bits specify the number of frames for transmission.

Range of allowable settings: Up to 4 Kframes ($0001_H$ to $1000_H$)

Do not make settings outside the range of allowable settings.

### 17.3.9 HSUSnRDAR — HS-USRT Data Destination Address Register

This register specifies the destination address of received data.

**Access:** This register can be read/written in 32-bit units.

**Address:** <HSUSn_base> + $0020_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSUSnRDAR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSUSnRDAR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.17 HSUSnRDAR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | HSUSn RDAR[31:0] | Receive Data Destination Address Bit<br>Destination Address of the receive data packet<br>Define the write pointer base address of the receive data packet. |

The value of this register can be changed while communication is halted (HSUSnEN.HSUSnEN bits = 0 and HSUSnCST.HSUSnACTF bits = 0).

#### HSUSnRDAR[31:0] Bits (Receive Data Destination Address)

These bits specify the destination address of receive data.

Since the received data write to the Global RAM (or Local RAM (PE)) is read in 32-frame units, the write value to the 6 lower-order bits is ignored, and reading these bits always returns 0.

## 17.3.10 HSUSnRPFM — HS-USRT Receive Frame number Register

This register specifies the number of receive frames.

**Access:** This register can be read/written in 32-bit units.

**Address:** <HSUSn_base> + $0024_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | HSUSnRPFM[18:16] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSUSnRPFM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.18 HSUSnRPFM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 18 to 0 | HSUSn RPFM[18:0] | Receive Frame Number Bits<br>Specify the number of receive frames.<br>Up to 256 K frames can be specified ($00001_H$ to $40000_H$).<br>Setting the value beyond above range is not allowed.<br>4 lines are used: The multiple number of 4 can be specified.<br>2 lines are used: The multiple number of 2 can be specified.<br>1 line is used: Any value can be specified. |

The value of this register can be changed while communication is halted (HSUSnEN.HSUSnEN bits = 0 and HSUSnCST.HSUSnACTF bits = 0).

### HSUSnRPFM[18:0] Bits (Receive Frame number)

These bits specify the number of receive frames.

Range of allowable settings: Up to 256 Kframes ($00001_H$ to $40000_H$)

Do not make settings outside the range of allowable settings.

## 17.3.11 HSUSnRGFM — HS-USRT Receive Group Frame Number Register

This register specifies the number of receive frames in a group.

**Access:** This register can be read/written in 32-bit units.

**Address:** <HSUSn_base> + $0028_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HSUSn RGFM[17:16] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSUSnRGFM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.19 HSUSnRGFM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 17 to 0 | HSUSnRGFM [18:0] | Receive Group Frame Number Bits<br>Specify the number of receive frames in a group.<br>Up to 192 K frames can be specified ($00000_H$ to $30000_H$).<br>Additional condition: HSUSnRGFM ≤ HSUSnRPFM<br>Setting the value beyond above range is not allowed.<br>4 lines are used: The multiple number of 4 can be specified.<br>2 lines are used: The multiple number of 2 can be specified.<br>1 line is used: Any value can be specified. |

The value of this register can be changed while communication is halted (HSUSnEN.HSUSnEN bits = 0 and HSUSnCST.HSUSnACTF bits = 0).

### HSUSnRGFM[17:0] Bits (Receive Group Frame number)

These bits specify the number of receive frames in a group.

If detection of receive frames in a group is not required, set $00000_H$ to this register.

Range of allowable settings: Up to 192 Kframes ($00000_H$ to $30000_H$); HSUSnRGFM ≤ HSUSnRPFM

Do not make settings outside the range of allowable settings.

## 17.3.12 HSUSnIREN — HS-USRT Interrupt Enable Register

This register enables or disables interrupt requests.

**Access:** This register is read/written in 32-bit units.

**Address:** <HSUSn_base> + $002C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | HSUSn OVEE | HSUSn PEE | HSUSn RPCE | HSUSn RGFE | HSUSn RFFE | HSUSn TPCE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 17.20 HSUSnIREN register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 | HSUSnOVEE | Overflow Error Interrupt Enable Bit<br>0: Disable Overflow Error interrupt<br>1: Enable Overflow Error interrupt |
| 4 | HSUSnPEE | Parity Error Interrupt Enable Bit<br>0: Disable Parity Error interrupt<br>1: Enable Parity Error interrupt |
| 3 | HSUSnRPCE | RX Completion Interrupt Enable Bit<br>0: Disable RX Completion interrupt<br>1: Enable RX Completion interrupt |
| 2 | HSUSnRGFE | RX of Group Frame Interrupt Enable Bit<br>0: Disable RX of Group Frame interrupt<br>1: Enable RX of Group Frame interrupt |
| 1 | HSUSnRFFE | RX of First Frame Interrupt Enable Bit<br>0: Disable RX of First Frame interrupt<br>1: Enable RX of First Frame interrupt |
| 0 | HSUSnTPCE | TX Completion Interrupt Enable Bit<br>0: Disable TX Completion interrupt<br>1: Enable TX Completion interrupt |

The value of this register can be changed while communication is halted (HSUSnEN.HSUSnEN bits = 0 and HSUSnCST.HSUSnACTF bits = 0).

### HSUSnTPCE Bit (Packet Transmission Complete Interrupt Enable)

This bit enables or disables output of the packet transmission complete interrupt request.

### HSUSnRFFE Bit (First Frame Reception Interrupt Enable)

This bit enables or disables output of the first frame reception interrupt request.

### HSUSnRGFE Bit (Group Frame Reception Interrupt Enable)

This bit enables or disables output of the group frame reception interrupt request.

#### HSUSnRPCE Bit (Packet Reception Complete Interrupt Enable)

This bit enables or disables output of the packet reception complete interrupt request.

#### HSUSnPEE Bit (Parity Error Interrupt Enable)

This bit enables or disables output of the parity error interrupt request.

#### HSUSnOVEE Bit (Overflow Error Interrupt Enable)

This bit enables or disables output of the overflow error interrupt request.

### 17.3.13 HSUSnIRST — HS-USRT Interrupt Status Register

This register indicates the interrupt status.

**Access:** This register is read-only in 32-bit units.

**Address:** <HSUSn_base> + $0030_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | HSUSn OVEF | HSUSn PEF | HSUSn RPCF | HSUSn RGFF | HSUSn RFFF | HSUSn TPCF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 17.21 HSUSnIRST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. |
| 5 | HSUSnOVEF | Overflow Error Interrupt Flag<br>0: No Overflow Error detected<br>1: Overflow Error detected |
| 4 | HSUSnPEF | Parity Error Interrupt Flag<br>0: No Parity Error detected<br>1: Parity Error detected |
| 3 | HSUSnRPCF | RX Completion Interrupt Flag<br>0: Receiving of frames is not complete<br>1: Receiving of frames is complete |
| 2 | HSUSnRGFF | RX of Group Frame Interrupt Flag<br>0: Receiving of group of frames is not complete<br>1: Receiving of group of frames is complete |
| 1 | HSUSnRFFF | RX of First Frame Interrupt Flag<br>0: Receiving of first frame not started<br>1: Receiving of first frame has started |
| 0 | HSUSnTPCF | TX Completion Interrupt Flag<br>0: Transmission of frames is not complete<br>1: Transmission of frames is complete |

If writing of 1 to the corresponding bit in the interrupt status clear register (HSUSnIRCL) to clear a flag is in contention with the condition for the flag becoming zero, one of the operations below will proceed.

- If the value of the flag is 0, the flag will be set.
- If the value of the flag is 1, the flag will be cleared.

#### HSUSnTPCF Flag (Packet Transmission Complete Flag)

This flag indicates that the transmission has been completed.

The packet transmission complete interrupt request (INT_TPC) is not output while this flag is 1.

When transfer is to be started, clear all interrupt status flags to 0 before enabling communications (setting the HSUSnEN bit to 1).

[Setting conditions]

- Transmission of the number of frames specified in the transmit frame number register (HSUSnTPFM) ing completed

[Clearing conditions]

- 1 being written to the packet transmission complete flag clear bit (HSUSnIRCL.HSUSnTPCC)
- Software reset being applied through the software reset register (HSUSnSRST)

### HSUSnRFFF Flag (First Frame Reception Flag)

This bit indicates that transmission from the Radar-ASIC has started.

The first frame reception interrupt request (INT_RFF) is not output while this flag is 1. When transfer is to be started, clear this flag to 0 before enabling communications (setting the HSUSnEN bit to 1).

[Setting condition]

- The first detection of the low level on the CSIO after reception has started

[Clearing conditions]

- 1 being written to the first frame reception flag clear bit (HSUSnIRCL.HSUSnRFFC)
- Software reset being applied through the software reset register (HSUSnSRST)

### HSUSnRGFF Flag (Group Frame Reception Flag)

This flag indicates that the first frame of a group has been detected.

The group frame reception interrupt request (INT_RGF) is not output while , overflow error flag (HSUSnOVEF) or this flag is 1. When transfer is to be started, clear all interrupt status flags to 0 before enabling communications (setting the HSUSnEN bit to 1).

[Setting conditions]

When the following two conditions are met:

- DMA transfer of the number of frames specified in the Rx group frame number register (HSUSnRGFM) being completed
- The HSUSnOVEF flag being 0

[Clearing conditions]

- 1 being written to the group frame reception flag clear bit (HSUSnIRCL.HSUSnRGFC)
- Software reset being applied through the software reset register (HSUSnSRST)

### HSUSnRPCF Flag (Packet Reception Complete Flag)

This flag indicates that the reception has been completed.

The packet reception complete interrupt request (INT_RPC) is not output while overflow error flag (HSUSnOVEF) or this flag is 1. When transfer is to started, clear all interrupt status flags to 0 before enabling communications (setting the HSUSnEN bit to 1).

[Setting conditions]

When the following two conditions are met:

- DMA transfer of the number of frames of received data specified in the Rx frame number register (HSUSnRPFM) being completed
- The HSUSnOVEF flag being 0

[Clearing conditions]

- 1 being written to the packet reception complete flag clear bit (HSUSnIRCL.HSUSnRPCC)
- Software reset being applied through the software reset register (HSUSnSRST)

#### HSUSnPEF Flag (Parity Error Flag)

This flag indicates that a parity error has been detected during reception.

While this flag is 1, the subsequent parity error interrupt request (INT_PE) is not output. When transfer is to be started, clear this flag to 0 before enabling communications (setting the HSUSnEN bit to 1).

[Setting conditions]

- A parity error being detected in the received data

[Clearing conditions]

- 1 being written to the parity error flag clear bit (HSUSnIRCL.HSUSnPEC)
- Software reset being applied through the software reset register (HSUSnSRST)

#### HSUSnOVEF Flag (Overflow Error Flag)

This flag indicates that the buffer has overflowed during reception.

While this flag is 1, the subsequent receive overflow error interrupt request (INT_OVE) is not output. When transfer is to be started, clear this flag to 0 before enabling transfer (setting the HSUSnEN bit to 1).

[Setting condition]

- The Rx buffer being overflowed

[Clearing conditions]

- 1 being written to the overflow error flag clear bit (HSUSnIRCL.HSUSnOVEC)
- Software reset being applied through the software reset register (HSUSnSRST)

## 17.3.14 HSUSnIRCL — HS-USRT Interrupt Status Clear Register

This register is used to clear the corresponding interrupt status flag.

**Access:** This register is written only in 32-, 16- or 8-bit units.

**Address:** <HSUSn_base> + $0034_H$

**Value after reset:** $0000\ 0000_H$ The read value of this register is always $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | HSUSn OVEC | HSUSn PEC | HSUSn RPCC | HSUSn RGFC | HSUSn RFFC | HSUSn TPCC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | W | W | W | W | W | W |

**Table 17.22 HSUSnIRCL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 | HSUSnOVEC | Overflow Error Interrupt Flag Clear Bit<br>0: No function<br>1: Clear the Overflow Error Flag (HSUSnIRST.HSUSnOVEF = 0) |
| 4 | HSUSnPEC | Parity Error Interrupt Flag Clear Bit<br>0: No function<br>1: Clear the Parity Error Flag (HSUSnIRST.HSUSnPEF = 0) |
| 3 | HSUSnRPCC | RX Completion Interrupt Flag Clear Bit<br>0: No function<br>1: Clear the RX Complete Flag (HSUSnIRST.HSUSnRPCF = 0) |
| 2 | HSUSnRGFC | RX of Group Frame Interrupt Flag Clear Bit<br>0: No function<br>1: Clear the RX of Group Frame Flag (HSUSnIRST.HSUSnRGFF = 0) |
| 1 | HSUSnRFFC | RX of First Frame Interrupt Flag Clear Bit<br>0: No function HSUSn<br>1: Clear the RX of First Frame Flag (HSUSnIRST.RFFF = 0) |
| 0 | HSUSnTPCC | TX Completion Interrupt Flag Clear Bit<br>0: No function<br>1: Clear the TX Complete Flag (HSUSnIRST.HSUSnTPCF = 0) |

When read, this register always returns $0000\ 0000_H$.

### HSUSnTPCC Bit (Packet Transmission Complete Flag Clear)

This bit clears the packet transmission complete flag (HSUSnIRST.HSUSnTPCF) to 0.

### HSUSnRFFC Bit (First Frame Reception Flag Clear)

This bit clears the first frame reception flag (HSUSnIRST.HSUSnRFFF) to 0.

### HSUSnRGFC Bit (Group Frame Reception Flag Clear)

This bit clears the group frame reception flag (HSUSnIRST.HSUSnRGFF) to 0.

#### HSUSnRPCC Bit (Packet Reception Complete Flag Clear)

This bit clears the packet reception complete flag (HSUSnIRST.HSUSnRPCF) to 0.

#### HSUSnPEC Bit (Parity Error Flag Clear)

This bit clears the parity error flag (HSUSnIRST.HSUSnPEF) to 0.

#### HSUSnOVEC Bit (Overflow Error Flag Clear)

This bit clears the overflow error flag (HSUSnIRST.HSUSnPEF) to 0.

## 17.3.15 HSUSnCST — HS-USRT Communication Status Register

This register indicates the communications status.

**Access:** This register can be read/ only in 32-bit units.

**Address:** <HSUSn_base> + $0038_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HSUSn ACTF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 17.23 HSUSnCST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | HSUSnACTF | Active Flag<br>Indicate the communication status of HS-USRT.<br>0: The communication is stopped<br>1: The communication is in operation |

### HSUSnACTF Flag (Active Flag)

This bit indicates the communications status of the HS-USRT.

[Clearing condition]

- Transfer is completed and the HS-USRT is stopped.

[Setting condition]

- Transfer is started by setting the HSUSnEN bit to 1

## 17.3.16 HSUSnCTFM — HS-USRT Current Transmit Frame Number Register

This register indicates the number of the current Tx frames.

**Access:** This register can be read only in 32-bit units.

**Address:** <HSUSn_base> + $003C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | HSUSnCTFM[12:0] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 17.24 HSUSnCTFM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is read. |
| 12 to 0 | HSUSn CTFM[12:0] | Current Transmit Frame Number Bit<br>Transmit Frame Counter Value<br>Indicates the number of the frames that were fetched from memory by DMA operation for transmission.<br>The data is always handled in blocks of 4 frames by DMA operation, and the value in the register may exceed that in the Transmit Frame Number Register (HSUSnTPFM), if HSUSnTPFM is not specifying a multiple of 4. |

### HSUSnCTFM[12:0] Bits (Current Tx Frame number)

These bits indicate the number of the current Tx frames.

This register indicates the number of frames that have been fetched from the Global RAM (or Local RAM) by DMA transfer.

The data is always read from the Global RAM (or Local RAM) in 4-frame units; accordingly the value of this register may exceed the value set in the HSUSnTPFMn register if the value of the Tx frame number register (HSUSnTPFM) is not a multiple of 4.

This register is cleared to $0000\ 0000_H$ after one cycle of PCLK has elapsed following the disabling of communications (by clearing the HSUSnEN bit to 0).

## 17.3.17 HSUSnCRFM — HS-USRT Current Receive Frame Number Register

This register indicates the number of the current Rx frames.

**Access:** This register can be read only in 32-bit units.

**Address:** <HSUSn_base> + $0040_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | HSUSnCRFM[18:16] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSUSnCRFM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 17.25 HSUSnCRFM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | When read, the value after reset is read. |
| 18 to 0 | HSUSn CRFM[18:0] | Current Receive Frame Number Bit<br>Receive Frame Counter Value<br>Indicate the number of the frames that were sent to memory by DMA operation from reception.<br>The data is always handled in blocks of 32 frames by DMA operation, and the value in the register may exceed that in the Receive Frame Number Register (HSUSnRPFM), if HSUSnRPFM is not specifying a multiple of 32. |

### HSUSnCRFM[18:0] Bits (Current Rx Frame number)

These bits indicate the number of the current Rx frames.

This register indicates the number of frames that have been written to the Global RAM (or Local RAM) by DMA transfer.

The data is always written to the Global RAM (or Local RAM) in 32-frame units; accordingly the value of this register may exceed the value set in the HSUSnRPFM register if the value of the Rx frame number register (HSUSnRPFM) is not a multiple of 32.

This register is cleared to $0000\ 0000_H$ after one cycle of PCLK has elapsed following the disabling of communications (by clearing the HSUSnEN bit to 0).

# 17.4 Operation

## 17.4.1 Overview of HS-USRT Operation

The HS-USRT is used for half-duplex clock synchronous serial transfer to and from the on-board radar system (radar ASIC). The HS-USRT supports communication in two modes, transmission and reception.

### 17.4.1.1 Operation of Transmit Mode

Transmit mode operates in the following order. For the setting procedure, see **Section 17.4.9, Setting Procedure**.

1. Set the communication enable bit (HSUSnEN) to 1.
2. Use the internal DMAC of the HS-USRT to read data for transmission from the address specified in the transmit data source address register (HSUSnTSAR) and store the data in the Tx buffer.
3. Transmission starts when the data for transmission is stored in the Tx block (serializer) from the Tx buffer.
4. In the transmission, the low level is output from the CSIO pin and a clock signal is output from the SCKIO pin. Data is output from the SDIOn pin in synchronization with the rising edge of the SCKIO clock signal (n = 0 to 3).
5. Upon completion of the transmission of the number of frames specified in the Tx frame number register (HSUSnTPFM), the high level is output from the CSIO pin, and this leads to output of the packet transmission complete interrupt request (INT_TPC).

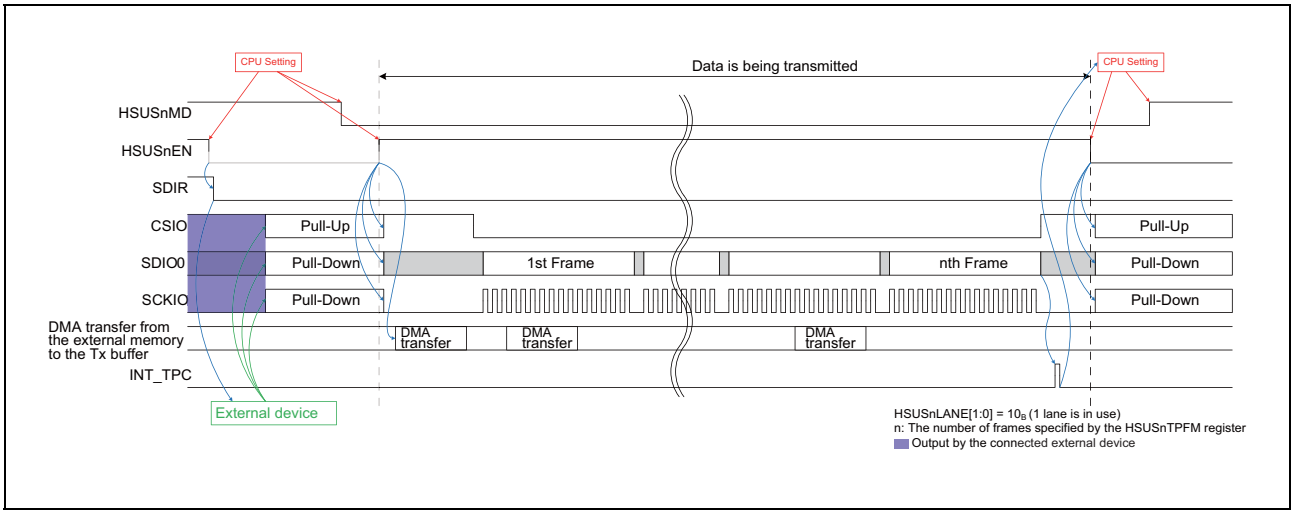



**Figure 17.2 Timing chart of transmission operation example**

### 17.4.1.2 Operation of Receive Mode

Receive mode operates in the following order. For the setting procedure, see **Section 17.4.9, Setting Procedure**.

1. Set the communication enable bit (HSUSnEN) to 1.
2. The high level is output from the CSIO pin.
3. The interface is ready for reception, which proceeds in synchronization with the Rx clock (SCKIO) while the CSIO pin is at the low level.
4. The Rx block (De-serializer) stores the received frame data in the Rx buffer.
5. DMA transfer to the address specified in the Rx data destination address register (HSUSnRDAR) starts when the received data is stored in the internal DMAC of the HS-USRT from the Rx buffer.
6. Upon completion of the DMA transfer of the number of frames of the received data specified in the Rx frame number register (HSUSnTPFM), the packet transmission complete interrupt request (INT_TPC) is output.
7. Setting the communication enable bit (HSUSnEN) to 0 leads to output of the low level from the SDIR pin.

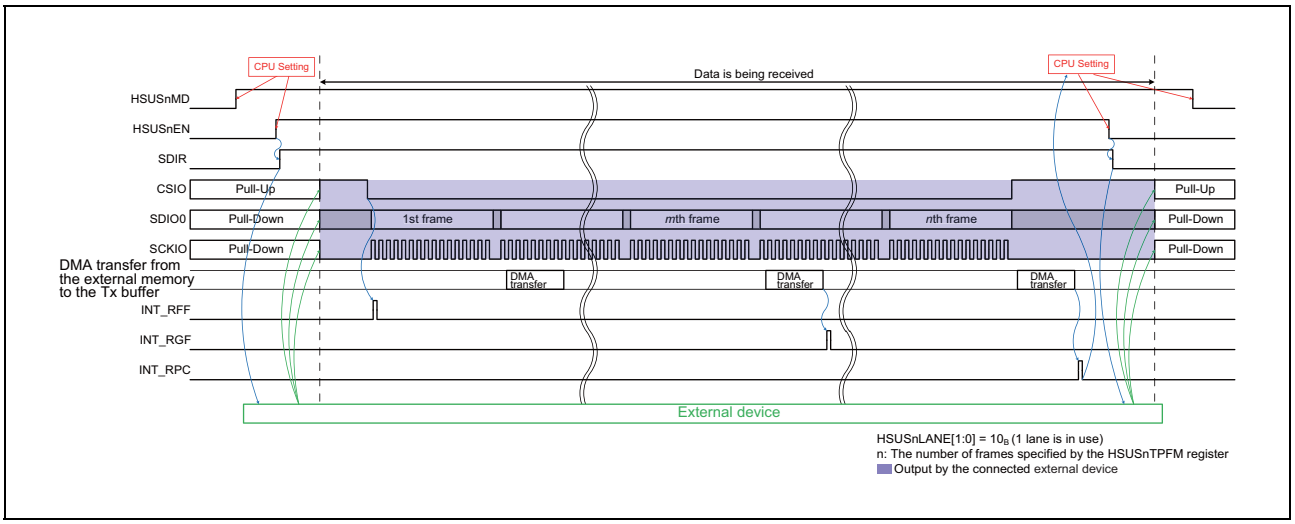



**Figure 17.3 Example of Reception Operation**

### 17.4.2 Control of the HS-USRT Pins

This table shows control of the HS-USRT pins.

**Table 17.26 Settings for Mode and Number of Lanes and Pin States and Operations**

| Mode | Pin | Pin State | | |
|---|---|---|---|---|
| Tx | Lines / Pin | Settings of HSUSnCTL.HSUSnLANE[1:0] | | |
| | | 4 lines ($00_B$) | 2 lines ($01_B$) | 1 lines ($10_B$) |
| | SDIO0 | Data output | Data output | Data output |
| | SDIO1 | Data output | Data output | Not used (low output) |
| | SDIO2 | Data output | Not used (low output) | Not used (low output) |
| | SDIO3 | Data output | Not used (low output) | Not used (low output) |
| | SDIR | Low level output | | |
| | CSIO | Output | | |
| | SCKIO | TX clock output | | |
| Rx | Lines / Pin | Settings of HSUSnCTL.HSUSnLANE[1:0] | | |
| | | 4 lines ($00_B$) | 2 lines ($01_B$) | 1 lines ($10_B$) |
| | SDIO0 | Data input | Data input | Data input |
| | SDIO1 | Data input | Data input | Not used (input disabled) |
| | SDIO2 | Data input | Not used (input disabled) | Not used (input disabled) |
| | SDIO3 | Data input | Not used (input disabled) | Not used (input disabled) |
| | SDIR | High level output | | |
| | CSIO | Input | | |
| | SCKIO | RX clock input | | |

**Table 17.27 Settings for Mode and Enable Bits and Pin Control**

| | Communication Mode Register (HSUSnMD) | Communication Enable Register (HSUSnEN) | Communication Direction Specification Pin | I/O Pin*1 |
|---|---|---|---|---|
| Mode | MD | EN | SDIR | CSIO, SCKIO, SDIOn |
| Tx | 0 | 0 | Low | Hi-Z*2 |
| | | 1 | Low | Output |
| Rx | 1 | 0 | Low | Hi-Z |
| | | 1 | High | Hi-Z (input) |

Note 1. Pull up the CSIO pin and pull down the SCKIO and SDIOn pins while the communication enable bit is 0, since this places the pins in the Hi-Z state.

Note 2. If the transmission is aborted by writing 0 to the HSUSnEN bit during transmission, the output state continues until transmission of the frame currently being transmitted is completed.

## 17.4.3 Data Format

### 17.4.3.1 Relationship between the Format of Data Storage and the Lines

This figure shows the format of data storage in memory and transfer of the data through the lines.

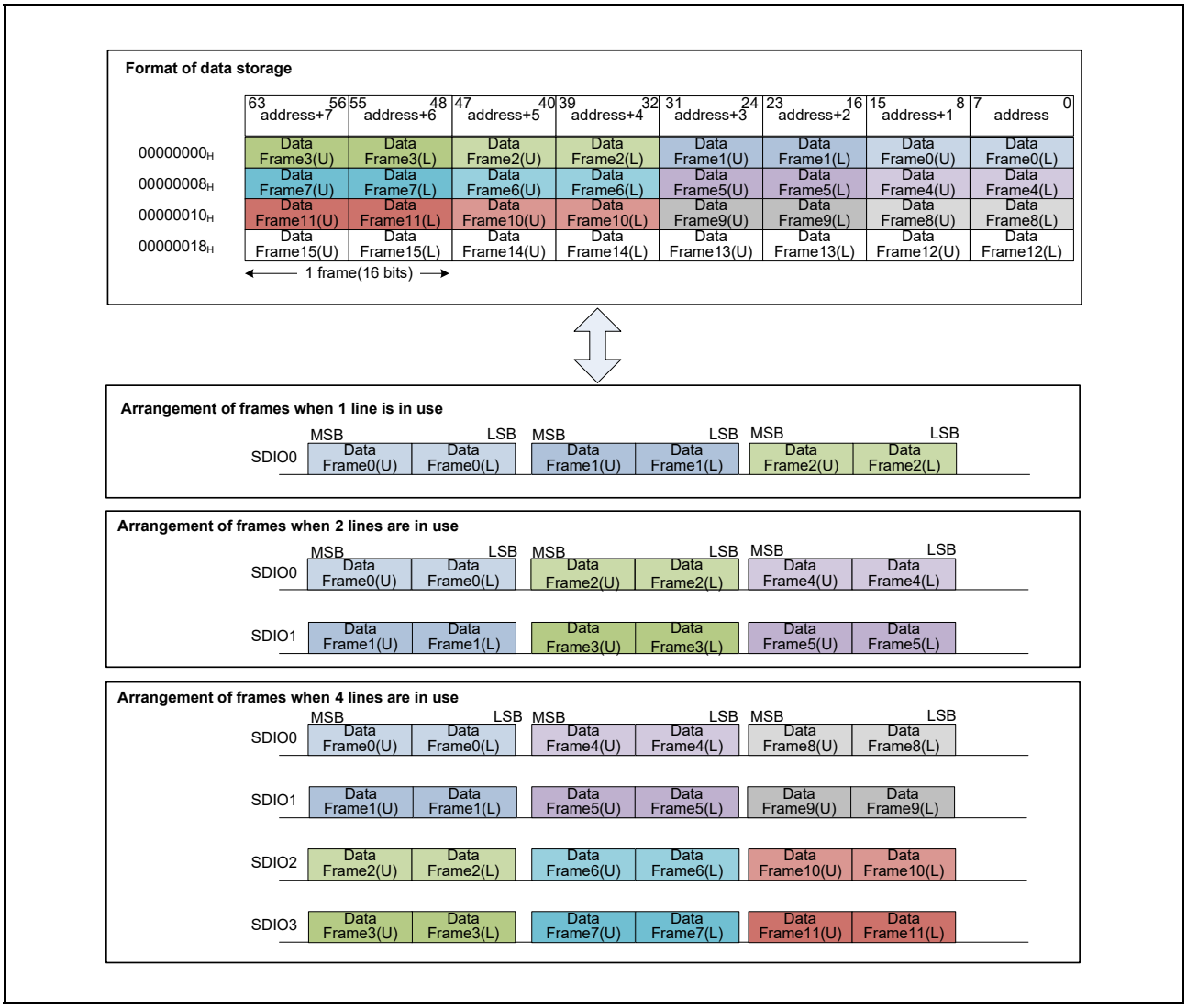



**Figure 17.4 Format of Data Storage in Memory and Transfer of the Data through the Lines**

### 17.4.3.2 Frame Data Format

This figure shows the format of frames. For details of parity, see **Section 17.4.3.3, Parity Control**.

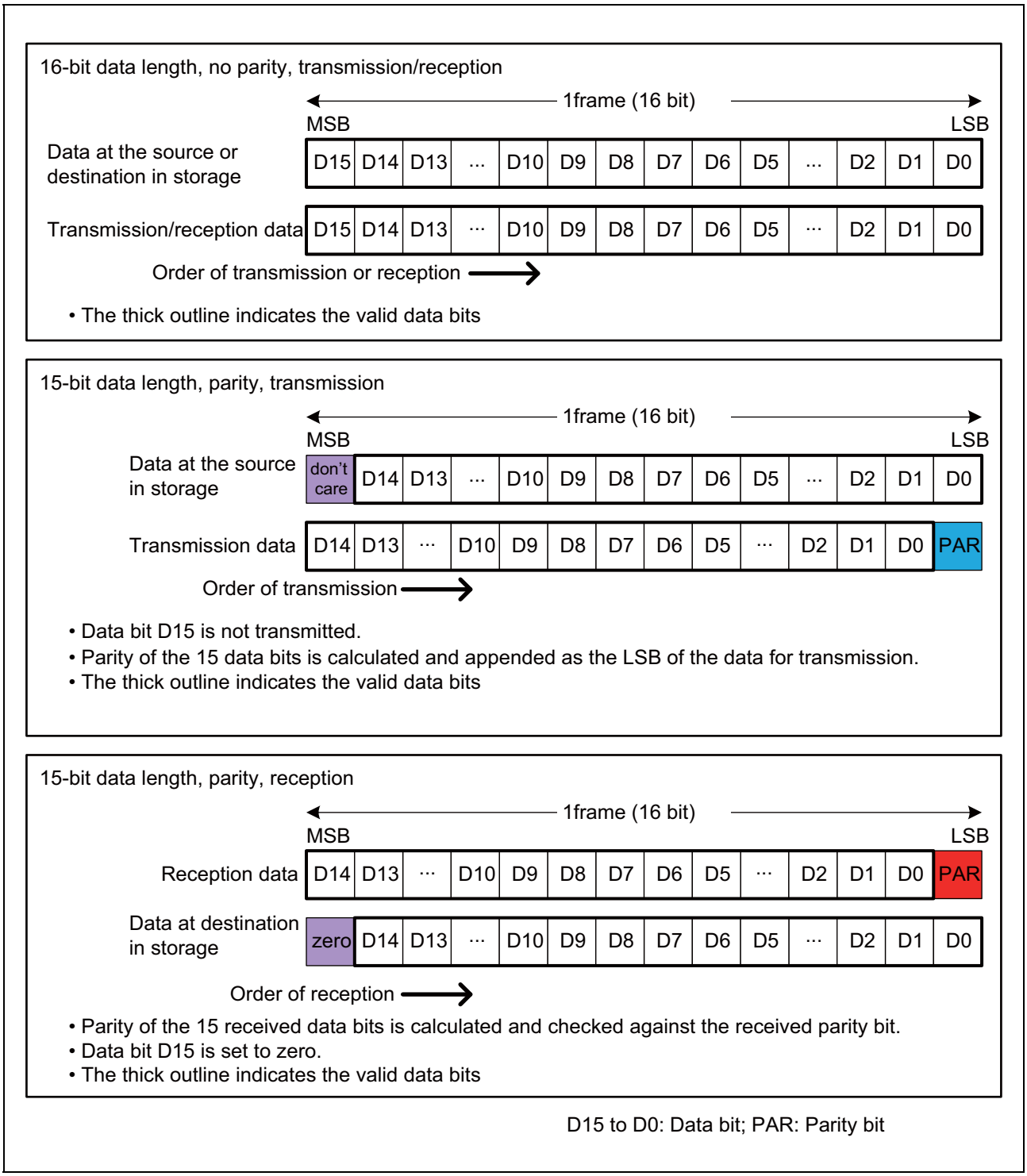



**Figure 17.5 Frame Format**

#### 17.4.3.3 Parity Control

A parity bit is used to detect bit errors in transferred data. Usually, the same type of parity will be used in transmission and reception.

Even parity and odd parity can be used to detect errors in 1 bit (and in other odd numbers of bits).

**(1) Even parity**

- For transmission
  The parity bit is controlled so that the number of 0-valued bits in the data for transmission, including the parity bit itself, is even. The value of the parity bit is as follows:
  Number of 1-valued bits in the data for transmission is odd: 1
  Number of 1-valued bits in the data for transmission is even: 0
- For reception
  The number of 1-valued bits in the received data, including the parity bit itself, is counted. If the result is an odd number, a parity error has occurred.

**(2) Odd parity**

- For transmission
  The parity bit is controlled so that the number of 0-valued bits in the data for transmission, including the parity bit itself, is odd. The value of the parity bit is as follows:
  Number of 1-valued bits in the data for transmission is odd: 0
  Number of 1-valued bits in the data for transmission is even: 1
- For reception
  The number of 1-valued bits in the received data, including the parity bit itself, is counted. If the result is an even number, a parity error has occurred.

**(3) No parity**

A parity bit is not appended to data for transmission.
Reception proceeds with no parity bit. Since received data do not include parity bits, parity errors will not occur.

#### 17.4.3.4 Relationship between the Clock Pin and Data

This figure shows the relationship between the level on the SCKIO pin and the timing of data output, and the timing of data output and sampling of data through the SDIO pins.

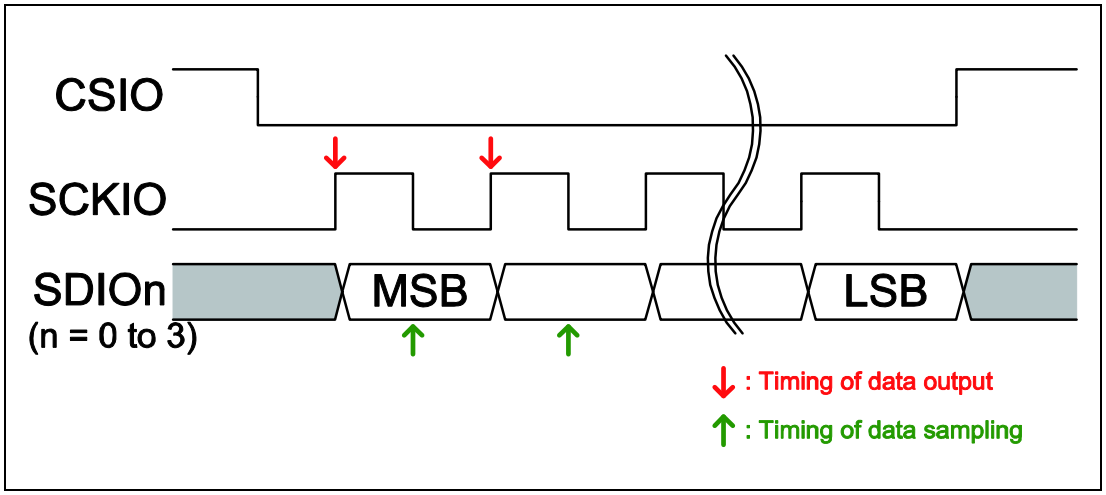



Figure 17.6 Relationship between the SCKIO Pin and Data

### 17.4.4 Buffer Configuration

#### 17.4.4.1 Configuration of Tx Buffers

Double buffering is used in transmission, so two buffers are provided. Each buffer can hold 4 frames of data for transmission.

When a buffer becomes empty, a request for the transfer of 4 frames of data for transmission is issued to the on-chip DMAC regardless of the settings of the bits for selecting the number of lanes (HSUSnLANE[1:0]) and the Tx frame number register (HSUSnTPFM).

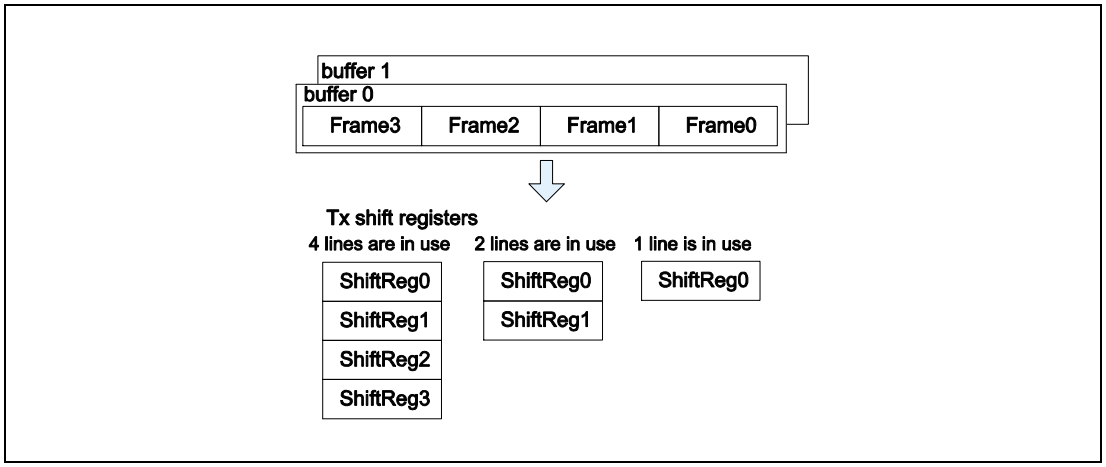



Figure 17.7 Configuration of Tx Buffers

### 17.4.4.2 Configuration of Rx Buffers

Double buffering is used in reception, so two buffers are provided. Each buffer can hold 32 frames (4 frames × 8) of received data.

When a buffer (buffer 0 or buffer 1) becomes full of received data, a request for transfer of the data is issued to the on-chip DMAC. The other buffer (buffer 1 or buffer 0) can continue reception during that time.

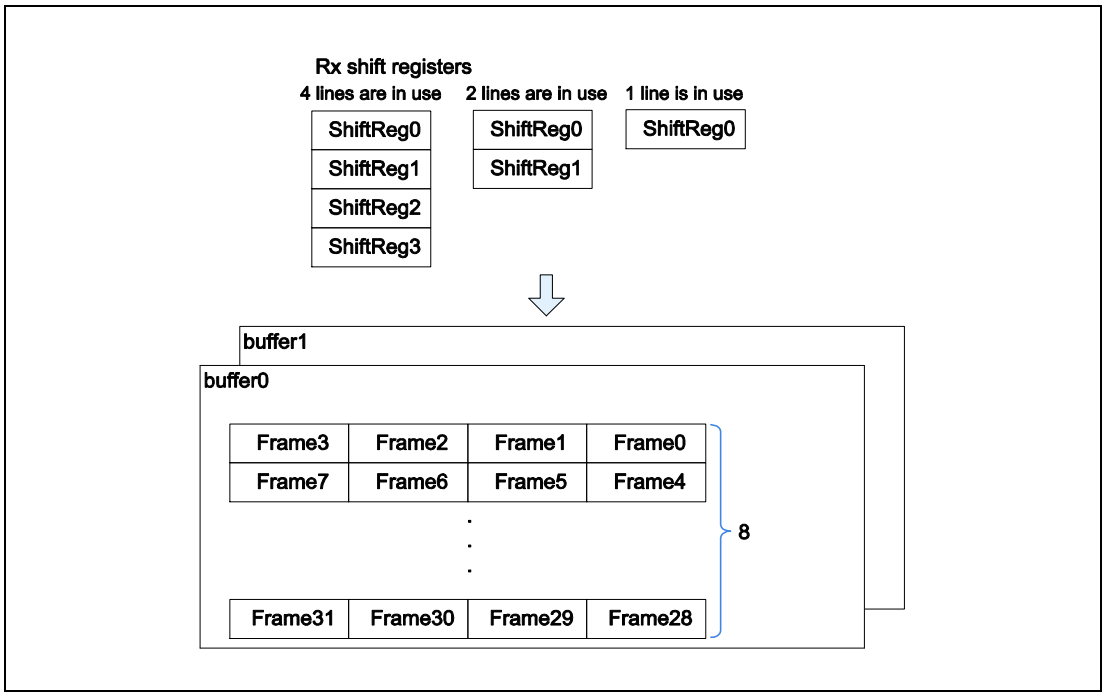



Figure 17.8 Configuration of Rx Buffers

## 17.4.5 Interrupt Requests

### 17.4.5.1 Interrupt Sources

The HS-USRT generates six types of interrupt request. Each interrupt source has its own interrupt request signal, and each interrupt request source has an enable bit, flag bit and flag clearing bit.

A flag is cleared by writing to the flag clearing bit. Writing 1 to the flag clearing bit for a flag bit that is indicating generation of the corresponding interrupt clears the flag bit.

The interrupt request signal is driven to the high level with a pulse width of one cycle of PCLK.

Each of the interrupt request signals is generated with the timing shown in **Figure 17.9, Timing of Interrupt Request Output**.

**Table 17.28 HS-USRT Interrupt Sources**

| Interrupt Name | Interrupt Sources | Symbol | Enable bit | Flag bit | Flag clear bit |
|---|---|---|---|---|---|
| INTHSUSnERR | Receive overflow error | INT_OVE | HSUSnOVEE | HSUSnOVEF | HSUSnOVEC |
| | Parity error | INT_PE | HSUSnPEE | HSUSnPEF | HSUSnPEC |
| INTHSUSnRE | First frame reception | INT_RFF | HSUSnRFFE | HSUSnRFFF | HSUSnRFFC |
| | Packet reception complete | INT_RPC | HSUSnRPCE | HSUSnRPCF | HSUSnRPCC |
| | Group frame reception | INT_RGF | HSUSnRGFE | HSUSnRGFF | HSUSnRGFC |
| INTHSUSnTR | Packet transmission complete | INT_TPC | HSUSnTPCE | HSUSnTPCF | HSUSnTPCC |

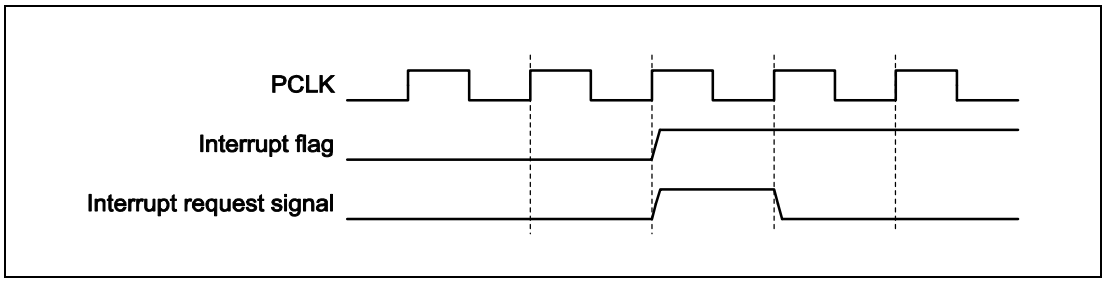



**Figure 17.9 Timing of Interrupt Request Output**

### 17.4.5.2 Packet Transmission Complete Interrupt Request

Upon completion of the transmission of the frames specified in the Tx frame number register (HSUSnTPFM), a packet transmission complete interrupt request (INT_TPC) is generated.

A packet transmission complete interrupt request (INT_TPC) is not generated while the packet transmission complete flag (HSUSnTPCF) is 1. When transfer is to be started, clear all interrupt status flags to 0 before enabling communications (setting the HSUSnEN bit to 1).

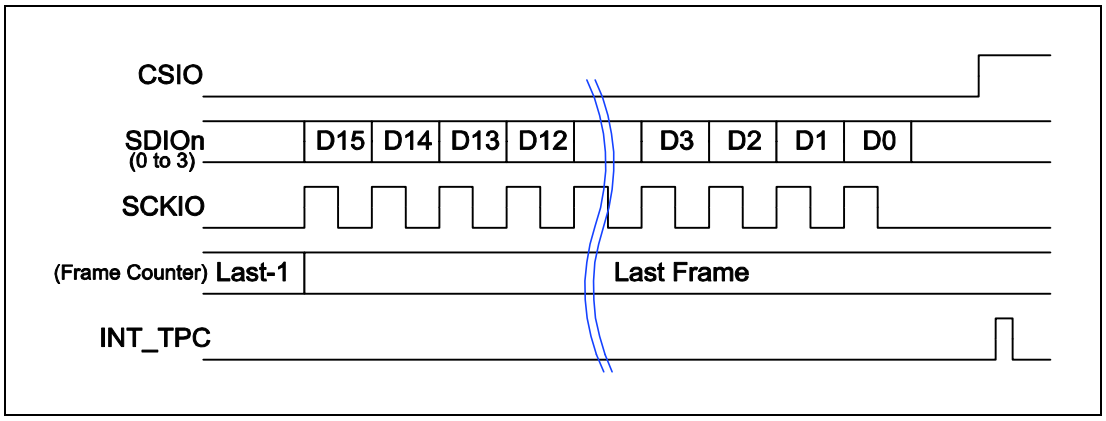



Figure 17.10 Packet Transmission Complete Interrupt Request

### 17.4.5.3 First Frame Reception Interrupt Request

When the low level on the CSIO pin is first detected after reception has started, a first frame reception interrupt request (INT_RFF) is generated.

The first frame reception interrupt request (INT_RFF) is not generated while the first frame reception flag (HSUSnRFFF) is 1. When transfer is to be started, clear the first frame reception flag (HSUSnRFFF) to 0 before enabling communications (setting the HSUSnEN bit to 1).

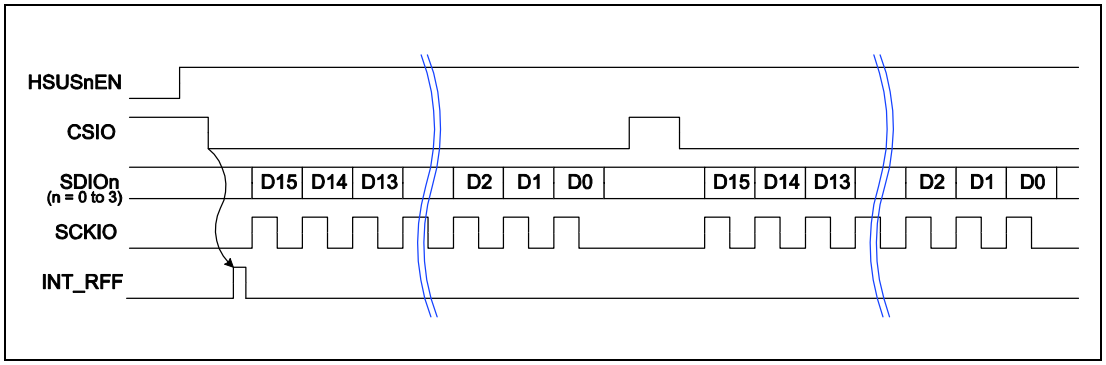



Figure 17.11 First Frame Reception Interrupt Request

### 17.4.5.4 Group Frame Reception Interrupt Request

If the value of the current Rx frame number register (HSUSnCRFM) has exceeded (or equal to) the value set in the Rx group frame number register (HSUSnRGFM) during reception, a group frame reception interrupt request (INT_RGF) is generated.

The group frame reception interrupt request (NT_RGF) is not generated while the Rx group frame number register (HSUSnRGFM) value is 0000 $0000_H$.

While the overflow error flag (HSUSnOVEF), or the group frame reception flag (HSUSnRGFF) is 1, the group frame reception interrupt request (INT_RGF) is not generated. When transfer is to be started, clear all interrupt status flags to 0 before enabling communications (setting the HSUSnEN bit to 1).

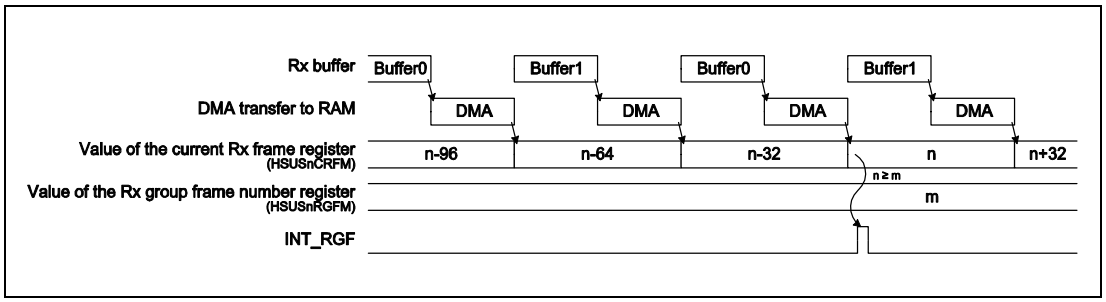



**Figure 17.12 Group Frame Reception Interrupt Request**

### 17.4.5.5 Packet Reception Complete Interrupt Request

If the value of the current Rx frame number register (HSUSnCRFM) has exceeded (or equal to) the value set in the Rx frame number register (HSUSnRPFM) during reception, a packet reception complete interrupt request (INT_RPC) is generated.

While the overflow error flag (HSUSnOVEF), or the packet reception complete flag (HSUSnRPCF) is 1, the packet reception complete interrupt request (INT_RPC) is not generated. When transfer is to be started, clear all interrupt status flags to 0 before enabling communications (setting the HSUSnEN bit to 1).

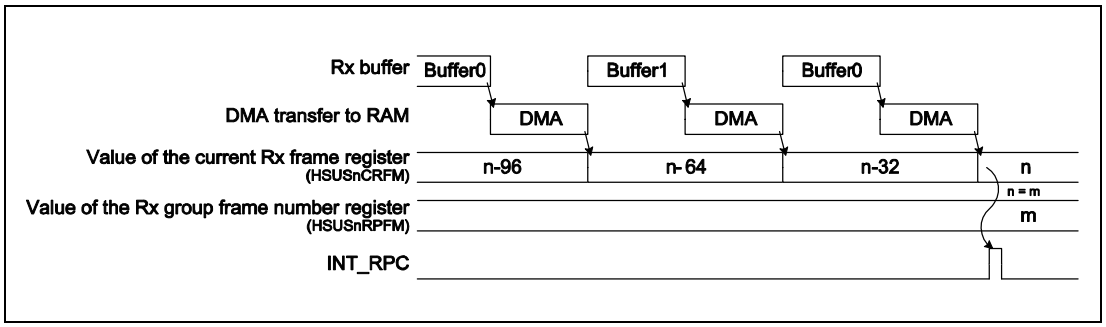



**Figure 17.13 Packet Reception Complete Interrupt Request**

#### 17.4.5.6 Parity Error Interrupt Request

If the parity value calculated from the received frame data and the value of the parity bit in the received data are not identical, a parity error interrupt request (INT_PE) is generated. The parity error interrupt request (INT_PE) is not generated while the parity error flag (HSUSnPEF) is 1 following the occurrence of a parity error interrupt. When transfer is to be started, clear the parity error flag (HSUSnPEF) to 0 before enabling communications (setting the HSUSnEN bit to 1).

The interrupt request is generated upon detection of a parity error in the Rx block (De-serializer).

Reception continues after a parity error.

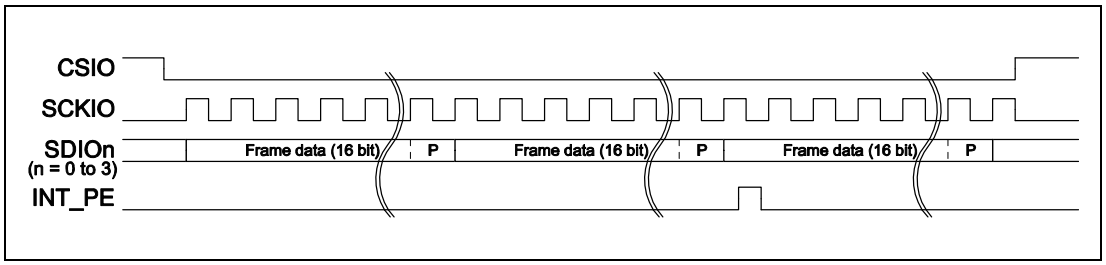



Figure 17.14 Parity Error Interrupt Request

#### 17.4.5.7 Receive Overflow Error Interrupt Request

Double buffering is also used in reception, so two buffers are provided. This allows the continuous reception of data by the interface while the DMA transfer of data that have already been received is proceeding.

If the received data is stored in the Rx buffer while the two buffers are full, a receive overflow error interrupt (INT_OVE) is generated. A group frame reception interrupt request (INT_RGF) or packet reception complete interrupt request (INT_RPC) is not generated while the overflow error flag (HSUSnOVEF) is 1 following the occurrence of a receive overflow error interrupt. Also, a receive overflow error interrupt (INT_OVE) is not generated. When transfer is to be started, clear the overflow error flag (HSUSnOVEF) to 0 before enabling communications (setting the HSUSnEN bit to 1).

If a receive overflow occurs, data in the buffer is overwritten.

### 17.4.6 Discarding Received Data

If the CSIO pin is driven to the high level while a frame is being received (before sampling of the 16th bit), the received data having fewer than 16 bits are judged to be invalid and the HS-USRT proceeds with the next operation. Note that generation of an interrupt request and indication by a status flag due to discarding of the received data do not proceed.

- Received data are discarded and not counted in the number of received frames.
- If a frame of received data judged to have been invalid is retransmitted and received normally, the retransmitted frame data is stored in the Rx buffer and counted as the number of received frames.
- If a frame of received data judged to have been invalid is not retransmitted, the value will not reach the number of received frames specified in the Rx frame number register (HSUSnRPFM), and the packet reception complete interrupt request will not be output. If a timeout is managed by a timer external to the HS-USRT, this will lead to a reception timeout.

### 17.4.7 Operation for Reception of Overrun Frames

If the radar ASIC transmits more frames than the number specified in the Rx frame number register (HSUSnRPFM), i.e. there is an overrun of frames, data in frames following the generation of the packet reception complete interrupt request (INT_RPC) are discarded.

If an overrun frame which has a parity error is received, the parity error interrupt request (INT_PE) is output.

In addition, if an overrun frame is received while the two Rx buffers are full, a receive overflow error interrupt (INT_OVE) is output.

### 17.4.8 Timeout

Following the start of reception after the CSIO pin is initially switched to the low level, a first frame reception interrupt request is generated to indicate the start of reception. This interrupt signal and a timer external to the HS-USRT can be used to detect a reception timeout error by monitoring whether packet reception is completed within a predetermined period.

**Figure 17.15** is an example where the external timer detects incomplete packet reception due to the discarding of received data while the second frame was being received.

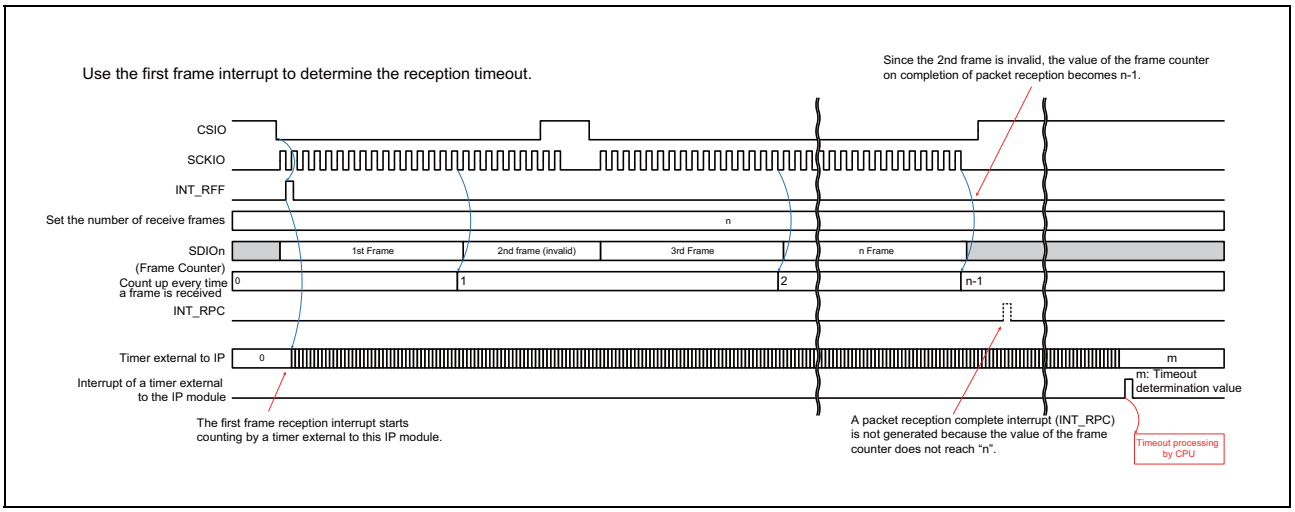



**Figure 17.15 Example of Timeout Error Detection**

## 17.4.9 Setting Procedure

### 17.4.9.1 Flows of Initialization and Stopping Operations

**Figure 17.16** shows the flow of initialization and **Figure 17.17** shows the flow of processing to stop operations after transfer is completed.

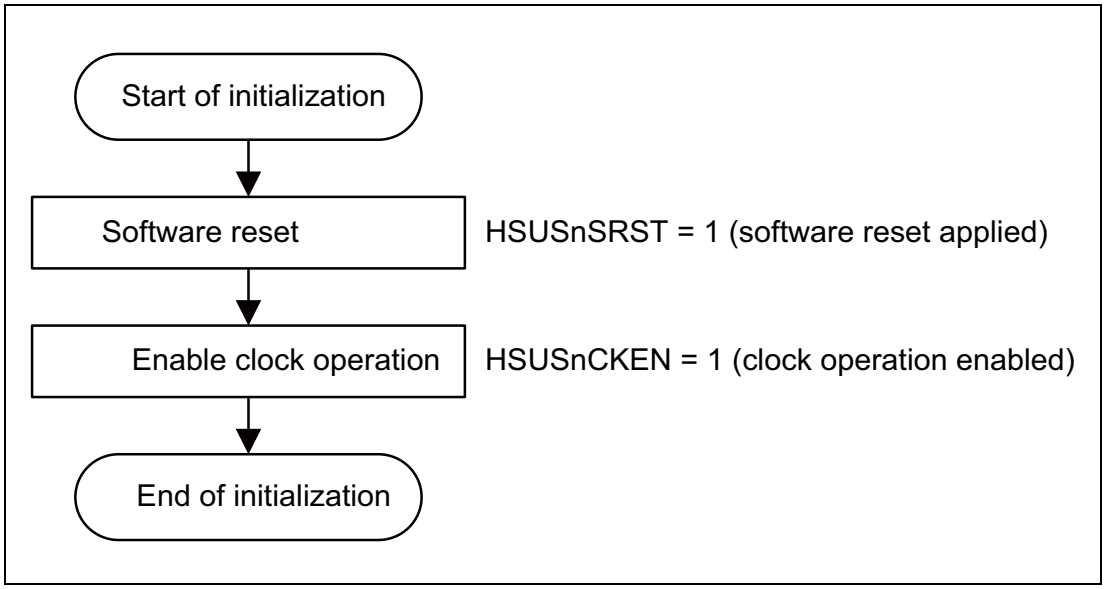



Figure 17.16 Initialization Flow

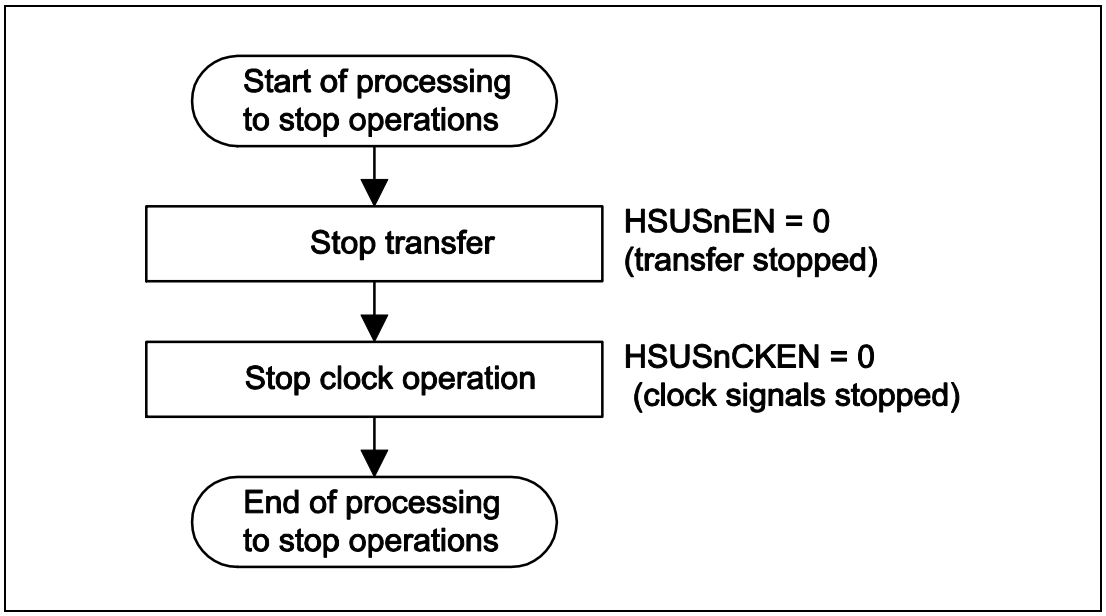



Figure 17.17 Flow of Processing to Stop Operations

### 17.4.9.2 Flow of Settings for Transmission

**Figure 17.18** shows an example of the flow of settings for transmission.

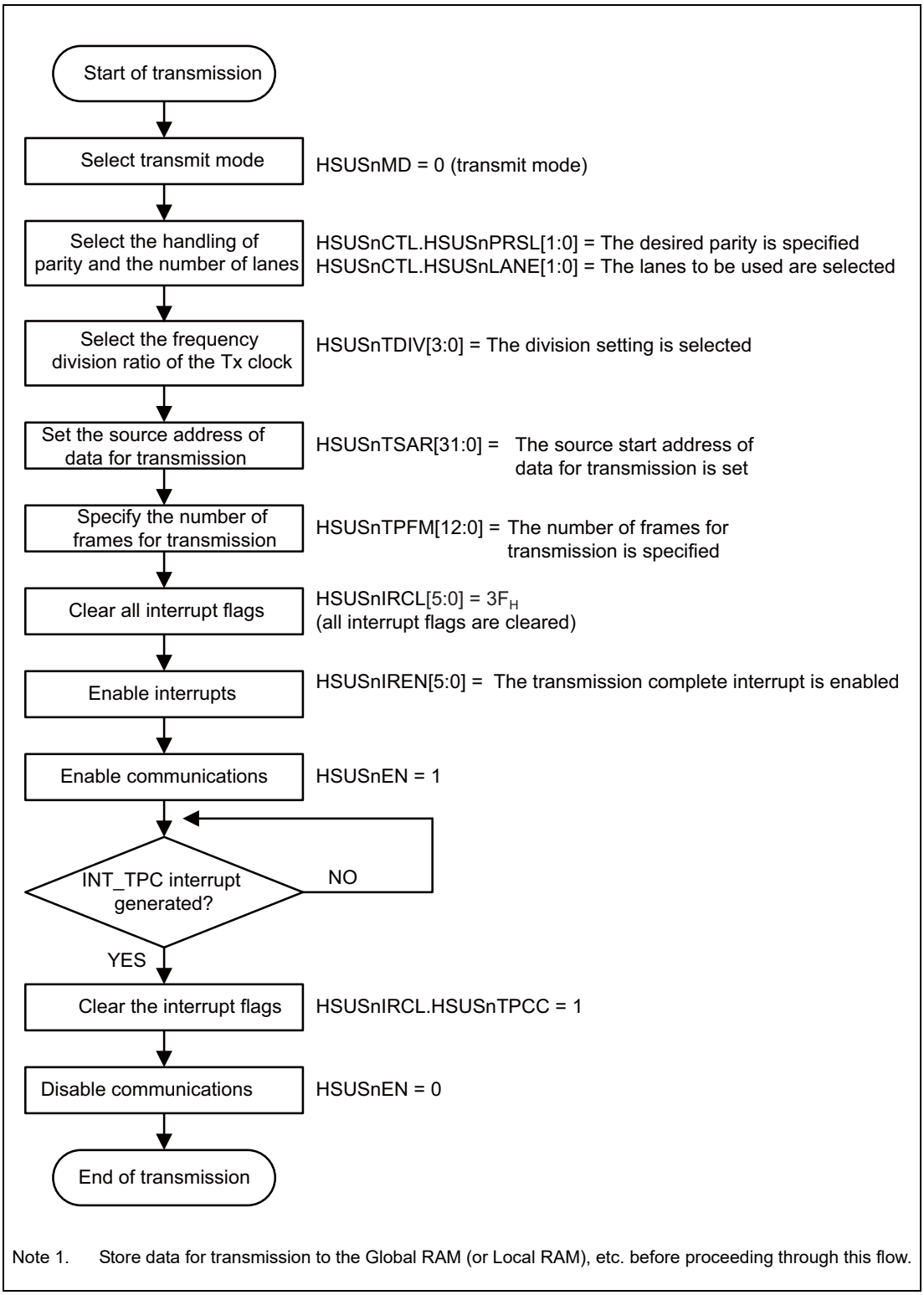



**Figure 17.18 Example of Transmission Setting Flow**

### 17.4.9.3 Flow of Settings for Reception

**Figure 17.19** shows an example of the flow of settings for reception.

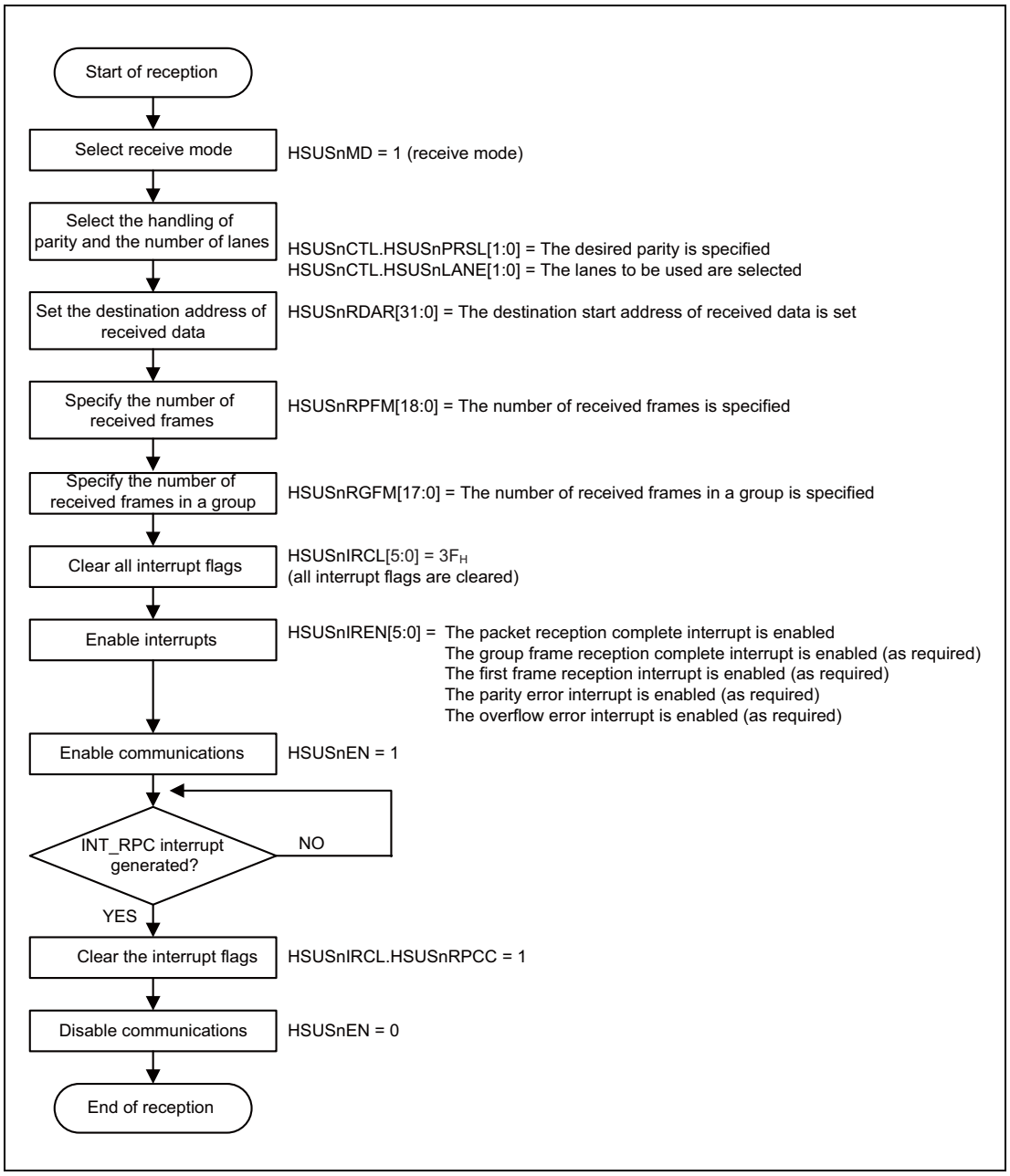



**Figure 17.19 Example of Reception Setting Flow**

#### 17.4.9.4 Flow of Error Processing

**Figure 17.20** shows an example of the flow of error processing.

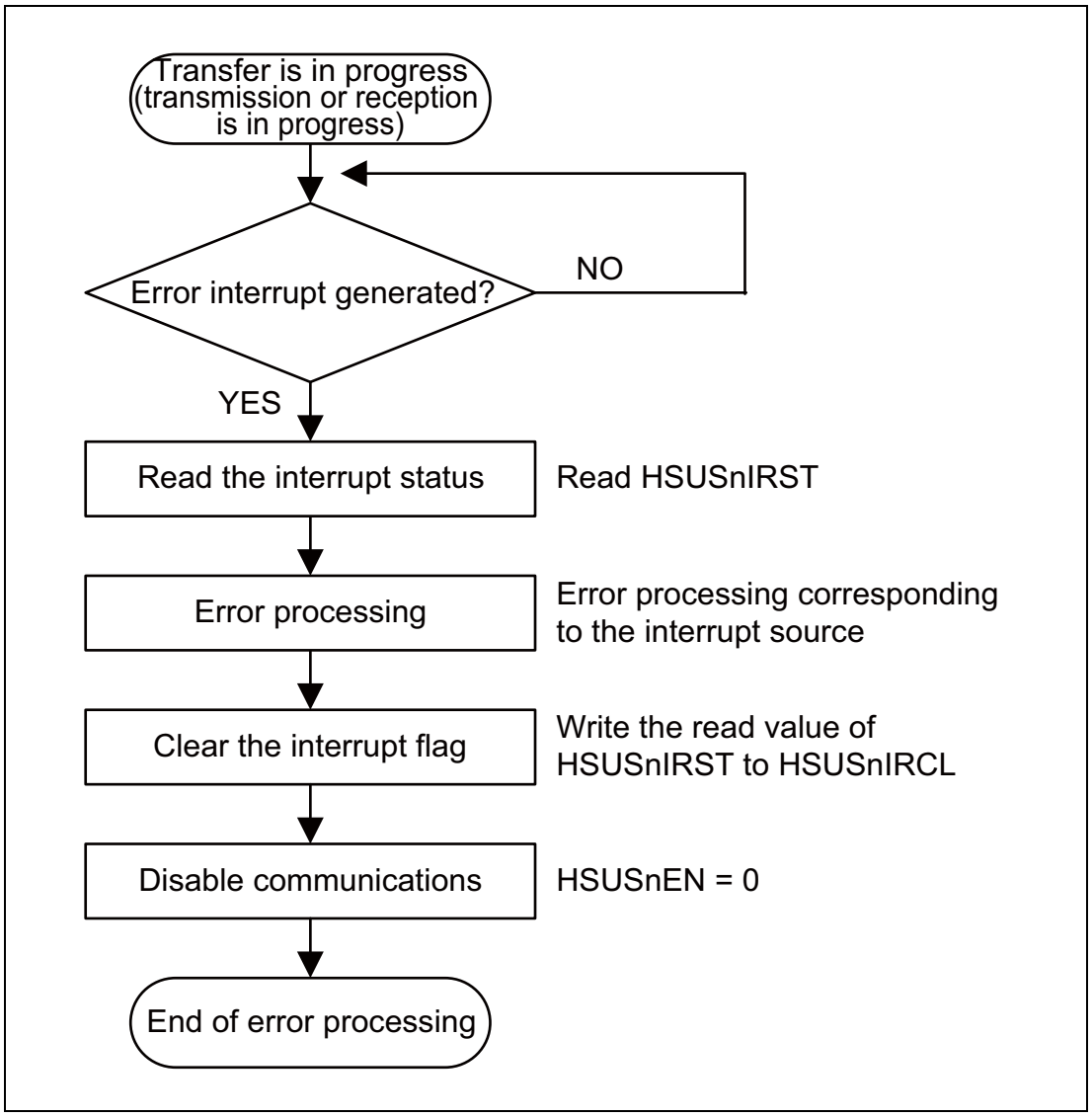



**Figure 17.20 Example of Error Processing Flow**

### 17.4.9.5 Aborting Transfer

Transfer is stopped by writing 0 to the HSUSnEN.HSUSnEN bit. Writing 0 to the HSUSnEN bit to stop transfer leads to the HSUSnCST.HSUSnACTF flag being cleared to 0, so read the latter register to confirm that transfer has actually been stopped, then apply a software reset.

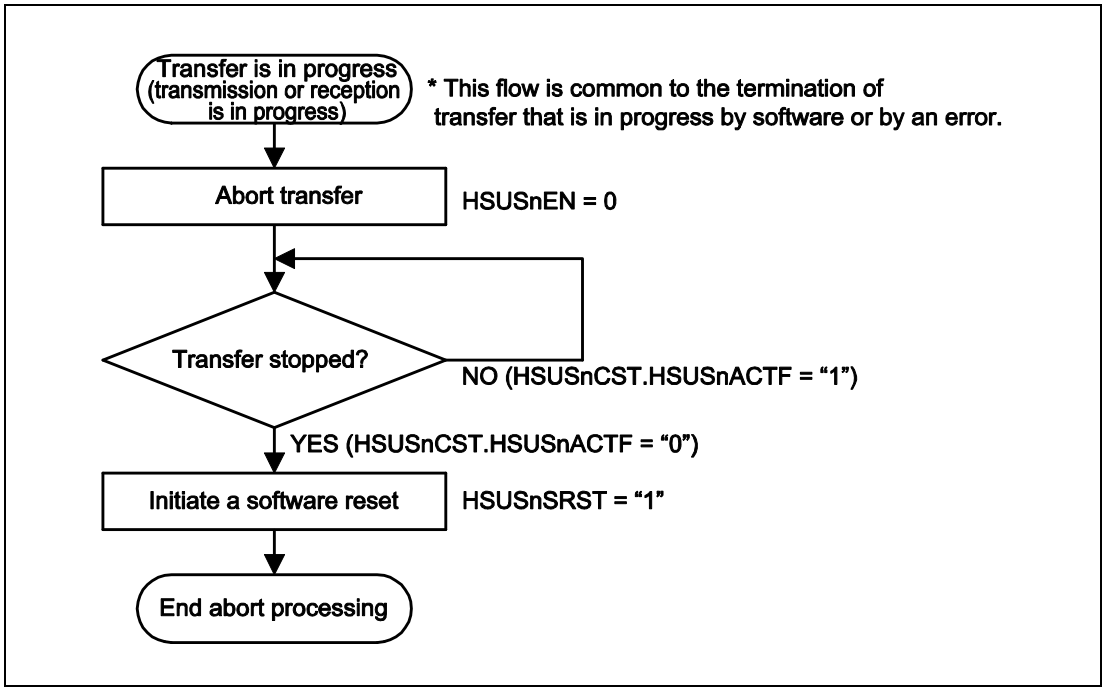



**Figure 17.21 Flow for Aborting Transfer**

**Table 17.29 Operations when Stopping Transfer in Progress**

| Communications mode | Operations when Stopping Transfer (by Writing 0 to HSUSnEN) |
|---|---|
| Transmission | On completion of transmission of the frame currently being transmitted, the CSIO pin is driven to the high level and transfer is stopped. |
| Reception | The frame being received is discarded and transfer is stopped. |

## 17.4.10 Notes

### 17.4.10.1 When Aborting Transfer

If transfer in progress is stopped, be sure to initiate a software reset. For details, see **Section 17.4.9.5, Aborting Transfer**

### 17.4.10.2 Write Protection for Setting Registers

Write protection for the setting registers is only applied while the communication enable bit (HSUSnEN) is 1 and the setting registers cannot be written to during this period. Write protection does not depend on the state of the active flag (HSUSnACTF), but if a software reset is applied while the active flag (HSUSnACTF) is 1, if transmission is in progress, it will immediately be aborted. Also, the H-bus transaction will be aborted.

Accordingly, when writing to the software reset register or the setting registers, do so while the communication enable bit (HSUSnEN) is 0 and the active flag (HSUSnACTF) is 0.

### 17.4.10.3 Clock Frequency

When the HS-USRT is to be used, make settings so that PCLK and HCLK have the same frequency and the relationship between TXCLK and HCLK is HCLK ≥ TXCLK. For reception, the relationship between SCKIO and HCLK must also satisfy the condition HCLK > SCKIO.

## 17.4.11 Limited Reset and Module stand-by

The HS-USRT can be reset by limited reset from SYSCTRL. The Limited Reset is functionally equivalent to a hardware reset. Software must ensure that HS-USRT is halted (HSUSnEN.HSUSnEN bit = 0 and HSUSnCST.HSUSnACTF bit = 0)

The HS-USRT clock can be disabled by the SYSCTRL module stand-by function. Software must ensure that HS-USRT is halted if module stand-by enable.

### 17.4.12 Data Storage and Address Space

The Transmission data in the Global RAM (or Local RAM) must set with the LSB end.

The internal DMAC of the HS-USRT store the receiving data in the Global RAM (or Local RAM) in 32-frame units as shown in **Figure 17.22**. Accordingly, if the number of received frames is not a multiple of 32, invalid data are written to the area that does not correspond to valid received frames, i.e. after the last frame of actual data is stored. Ensure that the storage area for received data has enough space so that such invalid data is not written over actual data.

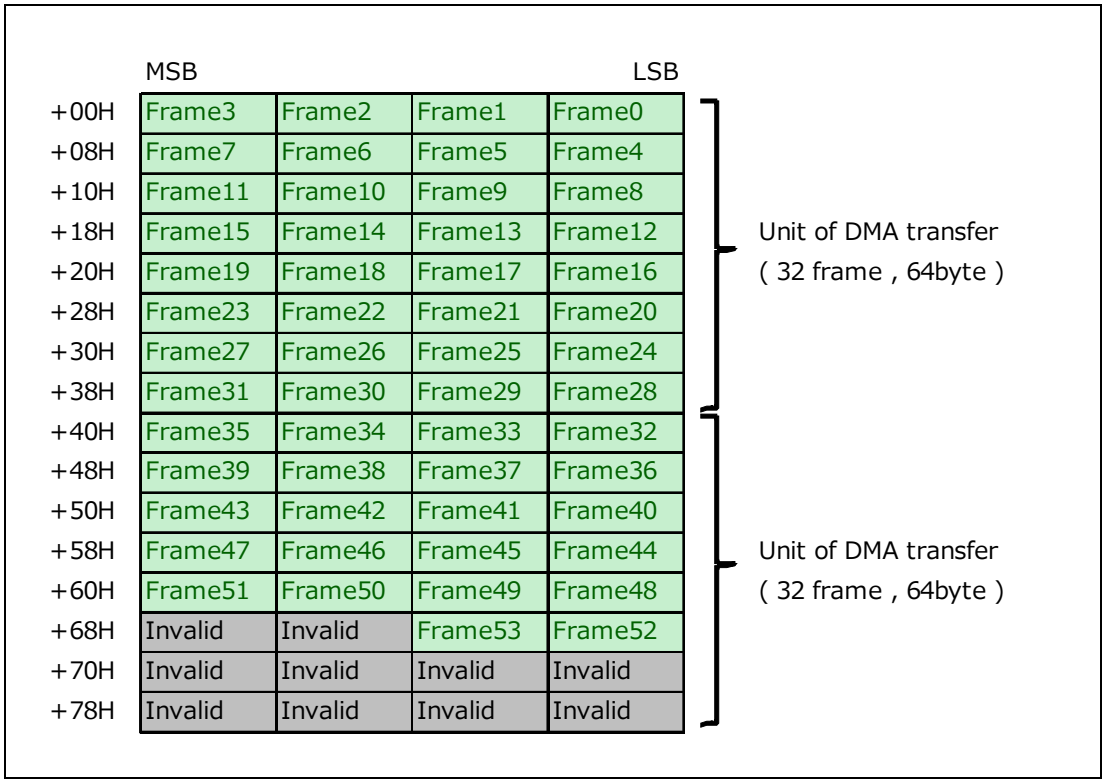



**Figure 17.22 Received Frame Data Storage Area**

## 17.5 Difference among P1M-C, P1H-C and P1H-CE

The difference among devices is only the number of units. For details, See **Table 17.1, HS-USRT Channels**.

Device P1M-C, P1H-C (4MB, BGA-156):

2 instances of HS_USRT (total receive bit rate is 320 Mbits/s)

Device P1H-C (4MB, BGA-292), P1H-C (8MB), P1H-CE:

4 instances of HS_USRT (total receive bit rate is 640 Mbits/s)

# Section 18 LIN/UART Interface (RLIN3)

This section contains a generic description of the LIN/UART interface (RLIN3).

The first part of this section describes all RH850/P1x-C specific properties, such as the number of units, register base addresses, etc.

The remainder of the section describes the functions and registers of RLIN3.

## 18.1 Features of RH850/P1x-C RLIN3

### 18.1.1 Units and Channels

This microcontroller has the following number of RLIN3 units.

RLIN3 unit has one channel interface. “Number of channels” is used with the same meaning as “number of units” in this section

**Table 18.1 Units**

| Product | P1M-C (QFP, BGA-292) | P1M-C (BGA-156) | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|
| Channel number(n) | 2 | 2 | 4 | 2 | 4 | 4 |
| Name | RLIN3n (n = 0 to 3) | | | | | |

**Table 18.2 Unit Configurations and Channels**

| Unit Name (Channel Name) RLIN3n | Channels per Unit | P1M-C (QFP, BGA-292) | P1M-C (BGA-156) | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|---|
| RLIN30 | 1 | √ | √ | √ | √ | √ | √ |
| RLIN31 | 1 | √ | √ | √ | √ | √ | √ |
| RLIN32 | 1 | — | — | √ | — | √ | √ |
| RLIN33 | 1 | — | — | √ | — | √ | √ |

**Note:** The channel names are same as those of the corresponding units.

The following lists the index corresponding to each product.

**Table 18.3 Index**

| Index | Meaning |
|---|---|
| n | Throughout this section, the individual RLN3 units are identified by the index “n” (n = 0 to 3): for example, RLN3nLCUC is the RLN3n control register. |
| b | Throughout this section, the individual transmit/receive buffers of RLN3n are identified by the index “b” (b = 1 to 8): for example, RLN3nLDBR1 is the first stage data buffer register. |

### 18.1.2 Register Base Address

RLIN3 base addresses are listed in the following table.

RLIN3 register addresses are given as offsets from the base addresses in general.

**Table 18.4 Register Base Address**

| Base Address Name | Base Address |
|---|---|
| <RLIN30_base> | FFD8 $C000_H$ |
| <RLIN31_base> | FFCA $C000_H$ |
| <RLIN32_base> | FFD8 $D000_H$ |
| <RLIN33_base> | FFCA $D000_H$ |

### 18.1.3 Clock Supply

Clock supply by and to RLIN3 is listed in the following table.

**Table 18.5 Clock Supply**

| Unit Name | Clock for the Unit | Supply Clock Name |
|---|---|---|
| RLIN3n | P-Bus interface clock (PCLK) | CLK_HSB |
| | RLIN3 communication clock (clkc) | CLKP_L |

**Note:** To use the RLIN3, following condition must be fulfilled: CLK_HSB ≥ CLKP_L

For detail of clock supply, see **Section 12, Clock Controller**.

### 18.1.4 Interrupt Request

RLIN3 interrupt requests are listed in the following table.

**Table 18.6 Interrupt Requests**

| Unit Interrupt Name | Outline | Interrupt Number | DMA Trigger Number | DTS Trigger Number (Primary) |
|---|---|---|---|---|
| **RLIN30** | | | | |
| INTRLIN3nUR0 (n = 0) | RLIN30 transmit interrupt | 104 | 91 | 101 |
| INTRLIN3nUR1 (n = 0) | RLIN30 receive completion interrupt | 103 | 90 | 100 |
| INTRLIN3nUR2 (n = 0) | RLIN30 status interrupt | 102 | — | — |
| **RLIN31** | | | | |
| INTRLIN3nUR0 (n = 1) | RLIN31 transmit interrupt | 107 | 93 | 103 |
| INTRLIN3nUR1 (n = 1) | RLIN31 receive completion interrupt | 106 | 92 | 102 |
| INTRLIN3nUR2 (n = 1) | RLIN31 status interrupt | 105 | — | — |
| **RLIN32** | | | | |
| INTRLIN3nUR0 (n = 2) | RLIN32 transmit interrupt | 110 | 95 | 105 |
| INTRLIN3nUR1 (n = 2) | RLIN32 receive completion interrupt | 109 | 94 | 104 |
| INTRLIN3nUR2 (n = 2) | RLIN32 status interrupt | 108 | — | — |
| **RLIN33** | | | | |
| INTRLIN3nUR0 (n = 3) | RLIN33 transmit interrupt | 113 | 97 | 107 |
| INTRLIN3nUR1 (n = 3) | RLIN33 receive completion interrupt | 112 | 96 | 106 |
| INTRLIN3nUR2 (n = 3) | RLIN33 status interrupt | 111 | — | — |

### 18.1.5 Reset sources

RLIN3 reset sources are listed in the following table. RLIN3 is initialized by these reset sources.

**Table 18.7 Register Reset Condition**

| | | Reset Condition | | | | |
|---|---|---|---|---|---|---|
| Unit name | Register Name | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| RLIN3n | All registers | √ | √ | √ | √ | √ |

**NOTE**

After reset, clear register value of error and status (RLN3nLMST, RLN3nLST and RLN3nLEST) and ignore UART reception data (RLIN3nLIDB, RLN3nLURDR, RLN3nLURDRL and RLN3nLURDRH).

### 18.1.6 External Input/Output Signals

External input/output signals of RLIN3 are listed below.

**Table 18.8 External Input/Output Signals**

| Unit Signal Name | Outline | Alternative port pin signal |
|---|---|---|
| **RLIN30** | | |
| RLIN3nRX (n = 0) | RLIN30 receive data input | RLIN30RX |
| RLIN3nTX (n = 0) | RLIN30 transmit data output | RLIN30TX |
| **RLIN31** | | |
| RLIN3nRX (n = 1) | RLIN31 receive data input | RLIN31RX |
| RLIN3nTX (n = 1) | RLIN31 transmit data output | RLIN31TX |
| **RLIN32** | | |
| RLIN3nRX (n = 2) | RLIN32 receive data input | RLIN32RX |
| RLIN3nTX (n = 2) | RLIN32 transmit data output | RLIN32TX |
| **RLIN33** | | |
| RLIN3nRX (n = 3) | RLIN33 receive data input | RLIN33RX |
| RLIN3nTX (n = 3) | RLIN33 transmit data output | RLIN33TX |

**Note:** RLIN30RX/TX is sharing terminal to three port pairs (MTTCAN0RX/MTTCAN0TX) of MTTCAN0 and one port pair of MCAN0.

# 18.2 Overview

## 18.2.1 Functional Overview

The LIN/UART interface is a hardware LIN communication controller that supports LIN Specification Package Revision 1.3, 2.0, 2.1, 2.2 and SAE J2602, and automatically performs frame communication and error determination.

The LIN/UART interface is provided with UART mode and can also be used as a UART.

The appropriate mode should be used for the LIN/UART interface according to the application: LIN master, LIN slave, or UART.

**LIN master**

- LIN reset mode
- LIN mode (LIN master mode)
  - LIN wake-up mode
  - LIN operation mode
- LIN self-test mode

**LIN slave**

- LIN reset mode
- LIN mode (LIN slave mode [auto baud rate] or LIN slave mode [fixed baud rate])
  - LIN wake-up mode
  - LIN operation mode
- LIN self-test mode

**UART**

- LIN reset mode
- UART mode

**Table 18.9** gives the LIN/UART interface specifications.

**Table 18.9 LIN/UART Interface Specifications (1/3)**

| Item | | Specifications | |
|---|---|---|---|
| | Channel count | 6 channels | |
| LIN communication function | Protocol | LIN Specification Package Revision 1.3, 2.0, and 2.1 | |
| | Variable frame structure | Master | • Break transmission width: 13 to 28 Tbits<br>• Break delimiter transmission width: 1 to 4 Tbits<br>• Inter-byte space (header): 0 to 7 Tbits (space between Sync field and ID field)*1<br>• Response space: 0 to 7 Tbits*1<br>• Inter-byte space: 0 to 3 Tbits (space between data bytes in response area)<br>• Wake-up: 1 to 16 Tbits |
| | | Slave | • Break reception width: 9.5 or 10.5 Tbits [for fixed baud rate]<br>: 10 or 11 Tbits [for auto baud rate]<br>• Response space: 0 to 7 Tbits<br>• Inter-byte space: 0 to 3 Tbits (space between data bytes in response area)<br>• Wake-up: 1 to 16 Tbits |
| | Checksum | • Automatic operation for both transmission and reception<br>• Classic or enhanced selectable (for each frame) | |
| | Response field data byte count | Variable from 0 to 8 bytes<br>Multi-byte (9 or more bytes) response transmission and reception also possible | |
| | Frame communication modes | Master | • Mode in which header transmission and response transmission/reception is started with a single transmission start request<br>• Mode in which header transmission and response transmission are started with separate transmission start requests (frame combined mode) |
| | | Slave | • Mode in which header is automatically received with fixed baud rate<br>• Mode in which header is automatically received with the baud rate set according to the sync field measurement result of the sync field and break field detected |
| | Wake-up transmission and reception | LIN wake-up mode provided<br>• Wake-up transmission (1 to 16 Tbits)<br>• Wake-up reception<br>Low-level width of input signals measured | |

Table 18.9 LIN/UART Interface Specifications (2/3)

| Item | | Specifications | |
|---|---|---|---|
| LIN communication function | Status | Master | • Successful frame/wake-up transmission<br>• Successful header transmission<br>• Successful frame/wake-up reception*2<br>• Successful 1st data byte reception<br>• Error detection<br>• Operation mode<br>(LIN reset mode, LIN wake-up mode, LIN operation mode, LIN self-test mode) |
| | | Slave | • Successful frame/wake-up transmission<br>• Successful frame/wake-up reception*2<br>• Successful header reception<br>• Successful 1st data byte reception<br>• Error detection<br>• Operation mode<br>(LIN reset mode, LIN wake-up mode, LIN operation mode, LIN self-test mode) |
| | Error status | Master | • Bit error<br>• Checksum error<br>• Frame timeout error/response timeout error<br>• Physical bus error<br>• Framing error<br>• Response preparation error |
| | | Slave | • Bit error<br>• Checksum error<br>• Frame timeout error/response timeout error<br>• Sync field error<br>• ID parity error<br>• Framing error<br>• Response preparation error |
| | Baud rate selection | Baud rate conforming to the LIN specifications generated using baud rate generator | |
| | Test mode | Self-test mode for user evaluation | |
| | Interrupt function | Master | • Successful header/frame/wake-up transmission<br>• Successful frame/wake-up reception*2<br>• Error detection |
| | | Slave | • Successful frame/wake-up transmission<br>• Header/frame/wake-up reception*2<br>• Error detection |

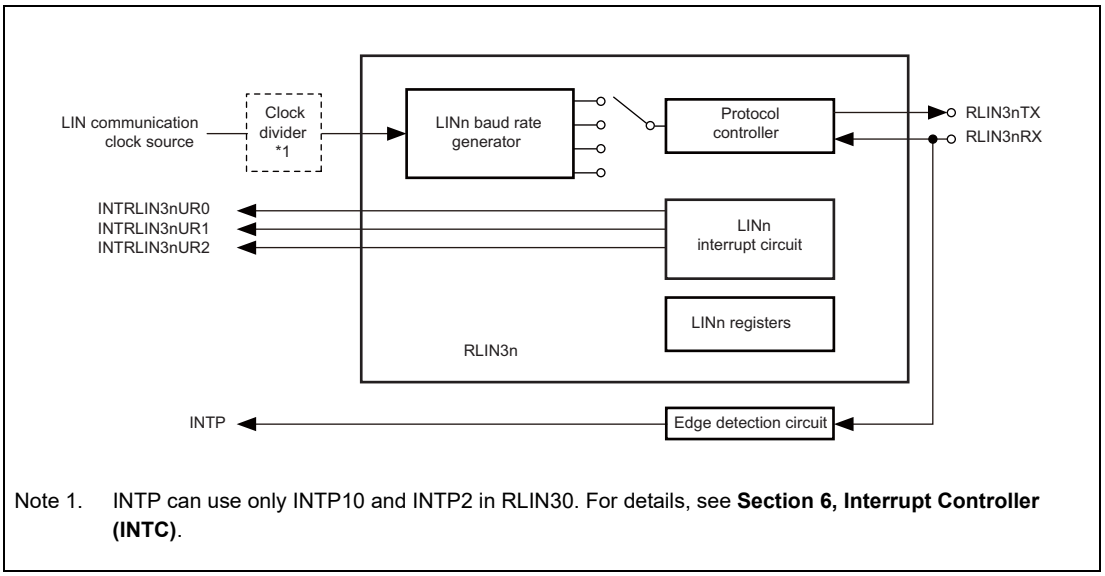
Table 18.9 LIN/UART Interface Specifications (3/3)

| Item | | Specifications |
|---|---|---|
| UART communication function | Data buffer | • Transmission data buffer/transmission data buffer for wait (exclusively for transmission; data length of 1. Character length of 7, 8, and 9 bits supported)<br>• UART multi-byte data (exclusively for transmission; variable data length from 1 to 9 bits; character length of 7 and 8 bits supported)<br>• Reception data buffer (exclusively for reception; data length of 1. Character length of 7, 8, and 9 bits supported) |
| | Data format | Character length: 7 or 8 bits<br>9 bits including the expansion bit supported. |
| | | Transmission stop bit: 1 or 2 bits |
| | | Parity function: odd, even, 0, or none |
| | | LSB- or MSB-first transfer selectable |
| | | Reverse input/output of transmission/reception data |
| | Status | • Transmission status<br>• Reception status<br>• Successful UART multi-byte data transmission<br>• Error SUM<br>• Expansion bit detection<br>• ID match<br>• Reset mode status |
| | Error status | • Bit error<br>• Framing error<br>• Parity error<br>• Overrun error |
| | Baud rate selection | With the baud rate generator incorporated, any baud rate can be set. |
| | When a certain expansion bit is at the expected level, the data received can be compared to the 8-bit data preset in the register. | |
| | The stop bit received is guaranteed (start of transmission can be delayed when start of transmission is attempted during reception of the stop bit). | |
| | Interrupt function | • Transmission start/complete<br>• Reception complete<br>• Status detection |

Note 1. Since the same register is used for setting, the inter-byte space (header) = response space.
Note 2. For wake-up reception, the low level width of the input signal is indicated.

### 18.2.2 Block Diagram

**Figure 18.1** shows a block diagram of the LIN/UART interface.




Note 1. INTP can use only INTP10 and INTP2 in RLIN30. For details, see **Section 6, Interrupt Controller (INTC)**.

**Figure 18.1 LIN/UART Interface Block Diagram**

### 18.2.3 Description of Blocks

- RLIN3nTX, RLIN3nRX: LIN/UART interface I/O pins
- LINn baud rate generator: Generates the LIN/UART interface communication clock signal.
- LINn registers: LIN/UART interface registers
- LINn interrupt controller: Controls interrupt requests generated by the LIN/UART interface

# 18.3 Registers

## 18.3.1 List of Registers

RLIN3 registers are listed in the following table.

For <RLIN3n_base>, see **Section 18.1.2, Register Base Address**.

**Table 18.10 Registers (1/2)**

| Module | Register | Symbol | Address | LIN Master | LIN Slave | UART | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | PBG | Other |
| RLN3n | LIN wake-up baud rate selector register | RLN3nLWBR | <RLIN3n_base> + $01_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART baud rate prescaler 01 register | RLN3n LBRP01 | <RLIN3n_base> + $02_H$ | — | √ | √ | *1 | — |
| RLN3n | LIN / UART baud rate prescaler 0 register | RLN3n LBRP0 | <RLIN3n_base> + $02_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART baud rate prescaler 1 register | RLN3n LBRP1 | <RLIN3n_base> + $03_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN self-test control register | RLN3nLSTC | <RLIN3n_base> + $04_H$ | √ | √ | — | *1 | — |
| RLN3n | LIN mode register | RLN3nLMD | <RLIN3n_base> + $08_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN break field configuration register/ UART configuration register | RLN3nLBFC | <RLIN3n_base> + $09_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART space configuration register | RLN3nLSC | <RLIN3n_base> + $0A_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN wake-up configuration register | RLN3nLWUP | <RLIN3n_base> + $0B_H$ | √ | √ | — | *1 | — |
| RLN3n | LIN interrupt enable register | RLN3nLIE | <RLIN3n_base> + $0C_H$ | √ | √ | — | *1 | — |
| RLN3n | LIN / UART error detection enable register | RLN3nLEDE | <RLIN3n_base> + $0D_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART control register | RLN3nLCUC | <RLIN3n_base> + $0E_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART transmission control register | RLN3nLTRC | <RLIN3n_base> + $10_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART mode status register | RLN3nLMST | <RLIN3n_base> + $11_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART status register | RLN3nLST | <RLIN3n_base> + $12_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART error status register | RLN3nLEST | <RLIN3n_base> + $13_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN data field configuration register | RLN3nLDFC | <RLIN3n_base> + $14_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART ID buffer register | RLN3nLIDB | <RLIN3n_base> + $15_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN checksum buffer register | RLN3nLCBR | <RLIN3n_base> + $16_H$ | √ | √ | — | *1 | — |
| RLN3n | UART data 0 buffer register | RLN3n LUDB0 | <RLIN3n_base> + $17_H$ | — | — | √ | *1 | — |
| RLN3n | LIN / UART data buffer 1 register | RLN3n LDBR1 | <RLIN3n_base> + $18_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART data buffer 2 register | RLN3n LDBR2 | <RLIN3n_base> + $19_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART data buffer 3 register | RLN3n LDBR3 | <RLIN3n_base> + $1A_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART data buffer 4 register | RLN3n LDBR4 | <RLIN3n_base> + $1B_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART data buffer 5 register | RLN3n LDBR5 | <RLIN3n_base> + $1C_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART data buffer 6 register | RLN3n LDBR6 | <RLIN3n_base> + $1D_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART data buffer 7 register | RLN3n LDBR7 | <RLIN3n_base> + $1E_H$ | √ | √ | √ | *1 | — |
| RLN3n | LIN / UART data buffer 8 register | RLN3n LDBR8 | <RLIN3n_base> + $1F_H$ | √ | √ | √ | *1 | — |
| RLN3n | UART operation enable register | RLN3n LUOER | <RLIN3n_base> + $20_H$ | — | — | √ | *1 | — |
| RLN3n | UART option register 1 | RLN3n LUOR1 | <RLIN3n_base> + $21_H$ | — | — | √ | *1 | — |

**Table 18.10 Registers (2/2)**

| Module | Register | Symbol | Address | LIN Master | LIN Slave | UART | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | PBG | Other |
| RLN3n | UART transmission data register | RLN3n LUTDR | <RLIN3n_base> + $24_H$ | — | — | √ | *1 | — |
| RLN3n | UART transmission data register L | RLN3n LUTDRL | <RLIN3n_base> + $24_H$ | — | — | √ | *1 | — |
| RLN3n | UART transmission data register H | RLN3n LUTDRH | <RLIN3n_base> + $25_H$ | — | — | √ | *1 | — |
| RLN3n | UART reception data register | RLN3n LURDR | <RLIN3n_base> + $26_H$ | — | — | √ | *1 | — |
| RLN3n | UART reception data register L | RLN3n LURDRL | <RLIN3n_base> + $26_H$ | — | — | √ | *1 | — |
| RLN3n | UART reception data register H | RLN3n LURDRH | <RLIN3n_base> + $27_H$ | — | — | √ | *1 | — |
| RLN3n | UART wait transmission data register | RLN3n LUWTDR | <RLIN3n_base> + $28_H$ | — | — | √ | *1 | — |
| RLN3n | UART wait transmission data register L | RLN3n LUWTDRL | <RLIN3n_base> + $28_H$ | — | — | √ | *1 | — |
| RLN3n | UART wait transmission data register H | RLN3n LUWTDRH | <RLIN3n_base> + $29_H$ | — | — | √ | *1 | — |

**Note:** √: Used
—: Not used
When writing to a register not used, write $00_H$.

Note 1. In the case of
n = 0 and 2 PBG3#0.PG3-RLIN3,
n = 1 and 3 PBG1#0.PG1-RLIN3

## 18.3.2 LIN Master Related Registers

### 18.3.2.1 RLN3nLWBR — LIN Wake-up Baud Rate Select Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $01_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NSPB[3:0] | | | | LPRS[2:0] | | | LWBR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.11 RLN3nLWBR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | NSPB[3:0] | Bit Sampling Count Select<br>b7 b4<br>0 0 0 0: 16 sampling<br>Settings other than the above are prohibited. |
| 3 to 1 | LPRS[2:0] | Prescaler Clock Select<br>b3 b1<br>0 0 0: 1/1<br>0 0 1: 1/2<br>0 1 0: 1/4<br>0 1 1: 1/8<br>1 0 0: 1/16<br>1 0 1: 1/32<br>1 1 0: 1/64<br>1 1 1: 1/128 |
| 0 | LWBR0 | Wake-up Baud Rate Select<br>0: When LIN1.3 is used.<br>1: When LIN2.x is used. |

Set the RLN3nLWBR register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### NSPB[3:0] Bits (Bit Sampling Count Select)

These bits select the number of sampling in one Tbit (reciprocal of the baud rate).
In LIN master mode (LIN/UART mode select bits in LIN mode register = $00_B$), set these bits to $0000_B$ (16 sampling).

#### LPRS[2:0] Bits (Prescaler Clock Select)

These bits select the frequency division ratio for the prescaler.
The LIN communication clock source is divided by this prescaler.

#### LWBR0 Bit (Wake-up Baud Rate Select)

When LIN Specification Package Revision 1.3 is used, set the LWBR0 bit in the RLN3nLWBR register to 0. This allows the 2.5-Tbit or longer low-level width of the input signal to be measured.

When LIN Specification Package Revision 2.x is used, set the LWBR0 bit to 1. Setting the LWBR0 bit to 1 selects the LIN system clock (fLIN) as fa regardless of the setting of the RLN3nLMD.LCKS bit (the LCKS bit is not changed). This allows the 2.5-Tbit or longer low-level width of the input signal to be measured.

Setting the baud rate to 19200 bps while fa is selected allows the 130 µs or longer low-level width of the input signal to be detected during LIN wake-up mode regardless of the setting of the RLN3nLMD.LCKS bit.

### 18.3.2.2 RLN3nLBRP0 — LIN Baud Rate Prescaler 0 Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $02_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LBRP0[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.12 RLN3nLBRP0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LBRP0[7:0] | Assuming that the value set in this register is N (0 to 255), the baud rate prescaler divides the frequency of the prescaler clock by N + 1.<br>Setting Range: $00_H$ to $FF_H$ |

Set the RLN3nLBRP0 register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

The value set in this register is used to control the frequency of baud rate clock sources fa, fb, and fc.

Assuming that the value set in this register is N, baud rate prescaler 0 divides the frequency of the clock that is selected by the LPRS bits by N + 1.

Registers RLN3nLBRP0 and RLN3nLBRP1 can be accessed as RLN3nLBRP01 in 16-bit units.

### 18.3.2.3 RLN3nLBRP1 — LIN Baud Rate Prescaler 1 Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $03_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LBRP1[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.13 RLN3nLBRP1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LBRP1[7:0] | Assuming that the value set in this register is M (0 to 255), the baud rate prescaler divides the frequency of the prescaler clock by M + 1.<br>Setting Range: $00_H$ to $FF_H$ |

Set the RLN3nLBRP1 register when the OMM0 bit in the RLN3LMST register is $0_B$ (in LIN reset mode).

The value set in this register is used to control the frequency of baud rate clock source fd.

Assuming that the value set in this register is M, baud rate prescaler 1 divides the frequency of the clock that is selected by the LPRS bits (prescaler clock select bits) by M + 1.

Registers RLN3nLBRP0 and RLN3nLBRP1 can be accessed as RLN3nLBRP01 in 16-bit units.

### 18.3.2.4 RLN3nLSTC — LIN Self-Test Control Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $04_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | LSTME[5:0] | | | | | | LSTM |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 18.14 RLN3nLSTC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value is undefined.<br>In writing to this bit, write value is meaningful only when $A7_H$, $58_H$, and $01_H$ are written successively to this register in LIN Self-Test mode. In other cases, write value is ignored. |
| 6 to 1 | LSTME[5:0] | LIN Self-Test Mode Entry<br>The test mode key values for configuring the RLIN3 module in Self-Test mode. |
| 0 | LSTM | LIN Self-Test Mode<br>0: The module is not in LIN self-test mode.<br>1: The module is in LIN self-test mode. |

The RLN3nLSTC register cancels protection of LIN self-test mode.

Set the RLN3nLSTC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

Writing $A7_H$, $58_H$, and $01_H$ successively to the RLN3nLSTC register places the module into LIN self-test mode.

When successive writing is completed thus placing LIN self-test mode to be entered, the LSTM bit is set to 1.

Do not write any other value during successive writing.

For making transition to LIN self-test mode, see **Section 18.5.5, LIN Self-Test Mode**.

When read, bits 6 to 1 return "000 $000_B$", and bit 7 returns an undefined value.

#### LSTM Bit (LIN Self-Test Mode)

When transition to LIN self-test mode is completed, the LSTM bit is set to 1.
For leaving LIN self-test mode, see **Section 18.5.5, LIN Self-Test Mode**.
Writing 1 to this bit does not affect the value of the RLN3nLSTC register if it is not a part of successive writing of $A7_H$, $58_H$, and $01_H$.

### 18.3.2.5 RLN3nLMD — LIN Mode Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base>+ $08_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | LRDNFS | LIOS | LCKS[1:0] | | LMD[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.15 RLN3nLMD register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 5 | LRDNFS | LIN Reception Data Noise Filtering Disable<br>0: The noise filter is enabled.<br>1: The noise filter is disabled. |
| 4 | LIOS | LIN Interrupt Output Select<br>0: Not supported, setting is prohibited.<br>1: RLIN3n transmission interrupt, RLIN3n successful reception interrupt, and RLIN3n status interrupt are used. |
| 3, 2 | LCKS[1:0] | LIN System Clock Select<br>b3 b2<br>0 0: fa (Clock generated by baud rate prescaler 0)<br>0 1: fb (1/2 clock generated by baud rate prescaler 0)<br>1 0: fc (1/8 clock generated by baud rate prescaler 0)<br>1 1: fd (1/2 clock generated by baud rate prescaler 1) |
| 1, 0 | LMD[1:0] | LIN/UART Mode Select<br>b1 b0<br>0 0: LIN master mode |

Set the RLN3nLMD register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode)

#### LRDNFS Bit (LIN Reception Data Noise Filtering Disable)

The LRDNFS bit enables or disables the noise filter when receiving data.
With 0 set, the noise filter is enabled when receiving data.
With 1 set, the noise filter is disabled when receiving data.

#### LIOS Bit (LIN Interrupt Output Select)

The LIOS bit selects the number of interrupt outputs from the LIN/UART interface.
With 0 set, the RLIN3 interrupt n is generated from the LIN/UART interface.
With 1 set, the RLIN3n transmission interrupt, RLIN3n successful reception interrupt, and RLIN3n status interrupt are generated from the LIN/UART interface.
For each interrupt source, see **Section 18.4, Interrupt Sources**.
The LIOS bit must always be set when using LIN mode operation, in order to enable the interrupt generation by the RLIN3 module. For UART mode, this bit is not relevant.

#### LCKS[1:0] Bits (LIN System Clock Select)

The LCKS bits select the clock to be input to the protocol controller.
With $00_B$ set, the protocol controller is provided with fa (clock generated by baud rate prescaler 0).
With $01_B$ set, the protocol controller is provided with fb (1/2 clock generated by baud rate prescaler 0).
With $10_B$ set, the protocol controller is provided with fc (1/8 clock generated by baud rate prescaler 0).
With $11_B$ set, the protocol controller is provided with fd (1/2 clock generated by baud rate prescaler 1).

With 1 is set in the LWBR0 bit in the RLN3nLWBR register (LIN 2.x is used), and the RLN3nLMST register is 01 (LIN wake-up mode), the protocol controller is provided with fa regardless of the setting of the bit (the LCKS bit is not changed)

#### LMD[1:0] Bits (LIN/UART Mode Select)

The LMD bits select the LIN/UART interface mode.
To use the LIN/UART interface as an LIN master, set these bits to $00_B$.

### 18.3.2.6 RLN3nLBFC — LIN Break Field Configuration Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $09_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | BDT[1:0] | | BLT[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.16 RLN3nLBFC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 5, 4 | BDT[1:0] | Transmission Break Delimiter (High Level) Width Select<br>b5 b4<br>0 0: 1 Tbit<br>0 1: 2 Tbits<br>1 0: 3 Tbits<br>1 1: 4 Tbits |
| 3 to 0 | BLT[3:0] | Transmission Break (Low Level) Width Select<br>b3 b0<br>0 0 0 0: 13 Tbits<br>0 0 0 1: 14 Tbits<br>0 0 1 0: 15 Tbits<br>:<br>1 1 1 0: 27 Tbits<br>1 1 1 1: 28 Tbits |

Set the RLN3nLBFC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

Some combinations of the set values result in the length of a frame exceeding the timeout time. Set the appropriate values in this register.

#### BDT[1:0] Bits (Transmission Break Delimiter (High Level) Width Select)

This bit is used to set the break high level width of transmission frame header.
1 Tbit to 4 Tbits can be set.

#### BLT[3:0] Bits (Transmission Break (Low Level) Width Select)

This BLT bits set the break low level width of transmission frame header.
13 Tbits to 28 Tbits can be set.

### 18.3.2.7 RLN3nLSC — LIN Space Configuration Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base>+ $0A_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | IBS[1:0] | | — | IBHS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R/W | R/W | R/W |

**Table 18.17 RLN3nLSC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 5, 4 | IBS[1:0] | Inter-Byte Space Select<br>b5 b4<br>0 0: 0 Tbit<br>0 1: 1 Tbit<br>1 0: 2 Tbits<br>1 1: 3 Tbits |
| 3 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 2 to 0 | IBHS[2:0] | Inter-Byte Space (Header)/Response Space Select<br>b2 b0<br>0 0 0: 0 Tbit<br>0 0 1: 1 Tbit<br>0 1 0: 2 Tbits<br>0 1 1: 3 Tbits<br>1 0 0: 4 Tbits<br>1 0 1: 5 Tbits<br>1 1 0: 6 Tbits<br>1 1 1: 7 Tbits |

Set the RLN3nLSC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

Some combinations of the set values result in the length of a frame or a response exceeding the timeout time. Set the appropriate values in this register.

#### IBS[1:0] Bits (Inter-Byte Space Select)

The IBS bits set the width of the inter-byte space of the transmission frame response field.
0 Tbit to 3 Tbits can be set.
These bits are enabled only during response transmission; these are disabled during response reception.

When data is transferred from the UART transmit data register (RLN3nLUTDR) and the UART wait transmit data register (RLN3nLUWTDR), the setting of these bits is ignored. Set these bits to "$00_B$"

#### IBHS[2:0] Bits (Inter-Byte Space (Header)/Response Space Select)

The IBHS bits set the width of the response space before response transmission.
0 Tbit to 7 Tbits can be set.
The response space setting is enabled only during response transmission; setting is disabled during response reception.
The inter-byte space (header) is equal to the response space.

### 18.3.2.8 RLN3nLWUP — LIN Wake-up Configuration Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base>+ $0B_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WUTL[3:0] | | | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R |

**Table 18.18 RLN3nLWUP register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | WUTL[3:0] | Wake-up Transmission Low Level Width Select<br>b7 b4<br>0 0 0 0: 1 Tbit<br>0 0 0 1: 2 Tbits<br>0 0 1 0: 3 Tbits<br>0 0 1 1: 4 Tbits<br>:<br>1 1 0 0: 13 Tbits<br>1 1 0 1: 14 Tbits<br>1 1 1 0: 15 Tbits<br>1 1 1 1: 16 Tbits |
| 3 to 0 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |

Set the RLN3nLWUP register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### WUTL[3:0] Bits (Wake-up Transmission Low Level Width Select)

The WUTL bits set the low level width of the wake-up signal transmission.
1 Tbit to 16 Tbits can be set.

With 1 is set in the LWBR0 bit in the RLN3nLWBR register (LIN 2.x is used), fa is selected as the LIN system clock (fLIN) regardless of the setting of the RLN3nLMD.LCKS bit (the LCKS bit is not changed).

### 18.3.2.9 RLN3nLIE — LIN Interrupt Enable Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $0C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SHIE | ERRIE | FRCIE | FTCIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 18.19 RLN3nLIE register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 3 | SHIE | Successful Header Transmission Interrupt Enable<br>0: Disables successful header transmission interrupt.<br>1: Enables successful header transmission interrupt. |
| 2 | ERRIE | Error Detection Interrupt Enable<br>0: Disables error detection interrupt.<br>1: Enables error detection interrupt. |
| 1 | FRCIE | Successful Frame/Wake-up Reception Interrupt Enable<br>0: Disables successful frame/wake-up reception interrupt.<br>1: Enables successful frame/wake-up reception interrupt. |
| 0 | FTCIE | Successful Frame/Wake-up Transmission Interrupt Enable<br>0: Disables successful frame/wake-up transmission interrupt.<br>1: Enables successful frame/wake-up transmission interrupt. |

Set the RLN3nLIE register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### SHIE Bit (Successful Header Transmission Interrupt Enable)

The SHIE bit enables or disables interrupt generation upon successful transmission of a header.
With 0 set, the interrupt is not generated when the HTRC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt is generated when the HTRC flag in the RLN3nLST register is set to 1.

#### ERRIE Bit (Error Detection Interrupt Enable)

The ERRIE bit enables or disables interrupt generation upon detection of an error.
With 0 set, the interrupt is not generated when the ERR flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt is generated when the ERR flag in the RLN3nLST register is set to 1.
Error types that are interrupt sources are the bit error, physical bus error, frame/response timeout error, framing error, checksum error, and response preparation error.
Detection of the bit error, physical bus error, frame/response timeout error, and framing error can be enabled or disabled using the RLN3nLEDE register.

### FRCIE Bit (Successful Frame/Wake-up Reception Interrupt Enable)

The FRCIE bit enables or disables interrupt generation upon successful reception of a frame or a wake-up signal (counting of low level width of the input signal).
With 0 set, the interrupt is not generated when the FRC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt is generated when the FRC flag in the RLN3nLST register is set to 1.

### FTCIE Bit (Successful Frame/Wake-up Transmission Interrupt Enable)

The FTCIE bit enables or disables interrupt generation upon successful transmission of a frame or a wake-up signal.
With 0 set, the interrupt is not generated when the FTC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt is generated when the FTC flag in the RLN3nLST register is set to 1.

### 18.3.2.10 RLN3nLEDE — LIN Error Detection Enable Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $0D_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LTES | — | — | — | FERE | FTERE | PBERE | BERE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R/W | R/W | R/W | R/W |

**Table 18.20 RLN3nLEDE register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | LTES | Timeout Error Select<br>0: Frame timeout error<br>1: Response timeout error |
| 6 to 4 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 3 | FERE | Framing Error Detection Enable<br>0: Disables framing error detection.<br>1: Enables framing error detection. |
| 2 | FTERE | Timeout Error Detection Enable<br>0: Disables frame/response timeout error detection.<br>1: Enables frame/response timeout error detection. |
| 1 | PBERE | Physical Bus Error Detection Enable<br>0: Disables physical bus error detection.<br>1: Enables physical bus error detection. |
| 0 | BERE | Bit Error Detection Enable<br>0: Disables bit error detection.<br>1: Enables bit error detection. |

Set the RLN3nLEDE register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode)

#### LTES Bit (Timeout Error Select)

The LTES bit selects the timeout function to be used.
With 0 set, the timeout function applies to frame timeout.
With 1 set, the timeout function applies to response timeout.
For details on the timeout error, see **Section 18.5.3.7, Error Status**.

#### FERE Bit (Framing Error Detection Enable)

The FERE bit enables or disables detection of the framing error.
With 0 set, the framing error is not detected.
With 1 set, the framing error is detected.
When this bit is set to 1, the detection result is indicated in the FER flag in the RLN3nLEST register.
For details on the framing error, see **Section 18.5.3.7, Error Status**.

#### FTERE Bit (Timeout Error Detection Enable)

The FTERE bit enables or disables detection of the frame timeout error or the response timeout error.
With 0 set, the frame timeout error or response timeout error is not detected.
With 1 set, the frame timeout error or response timeout error is detected.
When this bit is set to 1, the detection result is indicated in the FTER flag in the RLN3nLEST register.
With the LTES bit, either the frame timeout error or response timeout error can be selected.
Do not use the timeout error if response data of 9 bytes or more is to be transmitted or received.
For details on the timeout error, see **Section 18.5.3.7, Error Status**.
Timeout error should be disabled for data group communication.

#### PBERE Bit (Physical Bus Error Detection Enable)

The PBERE bit enables or disables detection of the physical bus error.
With 0 set, the physical bus error is not detected.
With 1 set, the physical bus error is detected.
When this bit is set to 1, the detection result is indicated in the PBER flag in the RLN3nLEST register.
For details on the physical bus error, see **Section 18.5.3.7, Error Status**.

#### BERE Bit (Bit Error Detection Enable)

The BERE bit enables or disables detection of the bit error.
With 0 set, the bit error is not detected.
With 1 set, the bit error is detected.
When this bit is set to 1, the detection result is indicated in the BER flag in the RLN3nLEST register.
For details on the bit error, see **Section 18.5.3.7, Error Status**.

### 18.3.2.11 RLN3nLCUC — LIN Control Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $0E_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | OM1 | OM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 18.21 RLN3nLCUC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 1 | OM1 | LIN Mode Select<br>0: LIN wake-up mode is caused.<br>1: LIN operation mode is caused. |
| 0 | OM0 | LIN Reset<br>0: LIN reset mode is caused.<br>1: LIN reset mode is canceled. |

Set the RLN3nLCUC register to $01_H$ to cause a transition to LIN wake-up mode after canceling LIN reset mode, and set the register to $03_H$ to cause a transition to LIN operation mode.

In LIN self-test mode, set the RLN3nLCUC register to 03h after a transition to LIN self-test mode is completed.

After a value is written to this register, confirm that the value written is actually indicated in the RLN3nLMST register before writing another value.

#### OM1 Bit (LIN Mode Select)

The OM1 bit selects the specific operation mode (either LIN wake-up mode or LIN operation mode) after canceling LIN reset mode.
With 0 set, LIN wake-up mode is caused.
With 1 set, LIN operation mode is caused.
This bit is valid only when the OMM0 bit in the RLN3nLMST register is 1.
Writing a value to this bit is disabled while the FTS bit in the RLN3nLTRC register is 1.

#### OM0 Bit (LIN Reset)

The OM0 bit selects either causing a transition to LIN reset mode or canceling LIN reset mode.
With 0 set, LIN reset mode is caused.
With 1 set, LIN reset mode is canceled.

### 18.3.2.12 RLN3nLTRC — LIN Transmission Control Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $10_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RTS | FTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 18.22 RLN3nLTRC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 1 | RTS | Response Transmission/Reception Start<br>0: Response transmission/reception is stopped in frame separate mode.<br>1: Response transmission/reception is started in frame separate mode. |
| 0 | FTS | Frame Transmission/Wake-up Transmission /Reception Start<br>0: Frame Transmission/wake-up transmission/reception is stopped.<br>1: Frame Transmission/wake-up transmission reception is started. |

#### RTS Bit (Response Transmission/Reception Start)

Set the RTS bit to 1 in frame separate mode after header transmission is started (FTS bit is 1) and response transmission data is ready. Once set, this bit is automatically cleared to 0 upon completion of frame communication or transition to LIN reset mode.
Only 1 can be written to this bit; 0 cannot be written.
To write 1 to this bit, write $02_H$ to the RLN3nLTRC register using the single memory storing instruction.
Writing a value to this bit is disabled when the OMM0 bit of the RLN3nLMST register is $0_B$ (in LIN reset mode).
Writing a value to this bit is disabled when the FTS bit is 0 (frame transmission or wake-up transmission/reception is halted).
When response data of 9 bytes or more is to be transmitted or received, set this bit to 1 each time a data group (variable from 0 to 8 bytes) is transmitted or received. Once set, this bit is automatically cleared to 0 at the end of communication or transition to LIN reset mode.

#### FTS Bit (Frame Transmission/Wake-up Transmission/Reception Start)

Set the FTS bit to 1 to start frame/wake-up transmission.
Also set this bit to 1 to allow wake-up reception (counting of the low level width of the input signal).
Only 1 can be written to this bit; 0 cannot be written.
Writing a value to this bit is disabled when the OMM0 bit of the RLN3nLMST register is $0_B$ (in LIN reset mode).

This bit is set to 0 upon completion of frame or wake-up communication and transition to LIN reset mode.

### 18.3.2.13 RLN3nLMST — LIN Mode Status Register (n = 0 to 3)

**Access:** This register can be read only in 8-bit units

**Address:** <RLIN3n_base> + $11_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | OMM1 | OMM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 18.23 RLN3nLMST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>Writing is ignored. |
| 1 | OMM1 | LIN Mode Status Monitor<br>0: The module is in LIN wake-up mode.<br>1: The module is in LIN operation mode. |
| 0 | OMM0 | LIN Reset Status Monitor<br>0: The module is in LIN reset mode.<br>1: The module is not in LIN reset mode. |

#### OMM1 Bit (LIN Mode Status Monitor)

The OMM1 bit indicates the current operating mode.

#### OMM0 Bit (LIN Reset Status Monitor)

The OMM0 bit indicates the current operating mode.

It takes 3 peripheral clock cycles + 4 communication clock cycles, until the OMM0 bit in the RLN3nLMST register reflects the OM0 bit setting in the RLN3nLCUC register.

### 18.3.2.14 RLN3nLST — LIN Status Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $12_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | HTRC | D1RC | — | — | ERR | — | FRC | FTC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R/W | R/W |

**Table 18.24 RLN3nLST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | HTRC | Successful Header Transmission Flag<br>0: Header transmission has not been completed.<br>1: Header transmission has been completed. |
| 6 | D1RC | Successful 1st data byte Reception Flag<br>These bits are always read as 0. The write value should always be 0. |
| 5, 4 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 3 | ERR | Error Detection Flag<br>0: No error has been detected.<br>1: Error has been detected. |
| 2 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 1 | FRC | Successful Frame/Wake-up Reception Flag<br>0: Frame or wake-up reception has not been completed.<br>1: Frame or wake-up reception has been completed. |
| 0 | FTC | Successful Frame/Wake-up Transmission Flag<br>0: Frame or wake-up transmission has not been completed.<br>1: Frame or wake-up transmission has been completed. |

The RLN3nLST register is automatically cleared to $00_H$ upon transition to LIN reset mode and start of the next communication (when the FTS bit of the RLN3nLTRC register is 1).

In LIN reset mode, this register cannot be written to. In LIN reset mode, the register retains $00_H$.

To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the single memory storing instruction.

#### HTRC Flag (Successful Header Transmission Flag)

Only 0 can be written to the HTRC flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The HTRC flag is set to 1 upon completion of header transmission. Here, an interrupt is generated if the SHIE bit in the RLN3nLIE register is 1 (interrupt is enabled). This bit is cleared automatically when LTRC.FTS is set.

### D1RC Flag (Successful 1st data byte Reception Flag)

Only 0 can be written to the D1RC flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The D1RC flag is set to 1 upon completion of 1st data byte reception. Here, an interrupt request is not generated. Here, an interrupt is not generated. This bit is cleared automatically when LTRC.FTS is set.
When response data of 9 bytes or more is to be received, this bit is set to 1 each time 1st data byte of a data group (variable from 0 to 8 bytes) is received. Write 0 before starting reception of the next data group.

### ERR Flag (Error Detection Flag)

The ERR flag is set to 1 upon detection of an error (when the value of at least one of the flags of the RLN3nLEST register is set to 1). Here, an interrupt is generated if the ERRIE bit in the RLN3nLIE register is 1 (interrupt is enabled). To clear the bit to 0 before the next communication (when the FTS bit of the RLN3nLTRC register is 1), write 0 to the RPER, CSER, FER, FTER, PBER, and BER flags in the RLN3nLEST register in LIN operation mode or LIN wake-up mode. This clears the ERR flag to 0.

### FRC Flag (Successful Frame/Wake-up Reception Flag)

Only 0 can be written to the FRC flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The FRC flag is set to 1 upon completion of frame or wake-up reception. Here, an interrupt is generated if the FRCIE bit in the RLN3nLIE register is 1 (interrupt is enabled). This bit is cleared automatically when LTRC.FTS is set.
When response data of 9 bytes or more is to be received, this bit is set to 1 each time a data group (variable from 0 to 8 bytes) is received. Write 0 before starting reception of the next data group.

### FTC Flag (Successful Frame/Wake-up Transmission Flag)

Only 0 can be written to the FTC flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The FTC flag is set to 1 upon completion of frame or wake-up transmission. Here, an interrupt is generated if the FTCIE bit in the RLN3nLIE register is 1 (interrupt is enabled). This bit is cleared automatically when LTRC.FTS is set.
When response data of 9 bytes or more is to be transmitted, this bit is set to 1 each time a data group (variable from 0 to 8 bytes) is transmitted. Write 0 before starting transmission of the next data group.

### 18.3.2.15 RLN3nLEST — LIN Error Status Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $13_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RPER | — | CSER | — | FER | FTER | PBER | BER |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R | R/W | R/W | R/W | R/W |

**Table 18.25 RLN3nLEST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RPER | Response Preparation Error Flag<br>0: Response preparation error has not been detected.<br>1: Response preparation error has been detected. |
| 6 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 5 | CSER | Checksum Error Flag<br>0: Checksum error has not been detected.<br>1: checksum error has been detected. |
| 4 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 3 | FER | Framing Error Flag<br>0: Framing error has not been detected.<br>1: Framing error has been detected. |
| 2 | FTER | Timeout Error Flag<br>0: Frame/response timeout error has not been detected.<br>1: Frame/response timeout error has been detected. |
| 1 | PBER | Physical Bus Error Flag<br>0: Physical bus error has not been detected.<br>1: Physical bus error has been detected. |
| 0 | BER | Bit Error Flag<br>0: Bit error has not been detected.<br>1: Bit error has been detected. |

The RLN3nLEST register is automatically cleared to $00_H$ upon transition to LIN reset mode and start of the next communication (when the FTS bit of the RLN3nLTRC register is 1).

In LIN reset mode, this register cannot be written to. In LIN reset mode, the register retains $00_H$.

When the FTS bit in the RLN3nLTRC register is 1 (frame transmission or wake-up transmission/ reception is started), do not write a value to this register.

To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the single memory storing instruction.

#### RPER Flag (Response Preparation Error Flag)

Only 0 can be written to the RPER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The RPER flag is set to 1 upon response preparation error detection. This happens, if response preparation was not completed before the first byte of a response has been received. This bit is cleared automatically when LTRC.FTS is set.

#### CSER Flag (Checksum Error Flag)

Only 0 can be written to the CSER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The CSER flag is set to 1 upon checksum error detection. This bit is cleared automatically when LTRC.FTS is set.

#### FER Flag (Framing Error Flag)

Only 0 can be written to the FER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
When the value of the FERE bit of the RLN3nLEDE register is 1 (framing error detection enabled), the FER flag is set to 1 upon framing error detection. This bit is cleared automatically when LTRC.FTS is set.

#### FTER Flag (Timeout Error Flag)

Only 0 can be written to the FTER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
When the FTERE bit of the RLN3nLEDE register is 1 (frame/response timeout error detection enabled), the FTER flag is set to 1 upon frame timeout error or response timeout error detection. This bit is cleared automatically when LTRC.FTS is set.

#### PBER Flag (Physical Bus Error Flag)

Only 0 can be written to the PBER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
When the PBERE bit of the RLN3nLEDE register is 1 (physical bus error detection enabled), the PBER flag is set to 1 upon physical bus error detection. This bit is cleared automatically when LTRC.FTS is set.

#### BER Flag (Bit Error Flag)

Only 0 can be written to the BER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
When the BERE bit of the RLN3nLEDE register is 1 (bit error detection enabled), the BER flag is set to 1 upon bit error detection. This bit is cleared automatically when LTRC.FTS is set.

### 18.3.2.16 RLN3nLDFC — LIN Data Field Configuration Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units.

**Address:** <RLIN3n_base> + $14_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LSS | FSM | CSM | RFT | RFDL[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.26 RLN3nLDFC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | LSS | Transmission/Reception Continuation Select<br>0: The data group to be transmitted/received next is the last one.<br>1: The data group to be transmitted/received next is not the last one. (Checksum is not included.) |
| 6 | FSM | Frame Separate Mode Select<br>0: Frame combined mode is set.<br>1: Frame separate mode is set. |
| 5 | CSM | Checksum Select<br>0: Classic checksum mode<br>1: Enhanced checksum mode |
| 4 | RFT | Response Field Communication Direction Select<br>0: Reception<br>1: Transmission |
| 3 to 0 | RFDL[3:0] | Response Field Length Select<br>b3 b0<br>0 0 0 0: 0 byte (+ checksum)<br>0 0 0 1: 1 byte (+ checksum)<br>0 0 1 0: 2 bytes (+ checksum)<br>:<br>0 1 1 1: 7 bytes (+ checksum)<br>1 0 0 0: 8 bytes (+ checksum)<br>1 0 0 1: 8 bytes (+ checksum)<br>1 0 1 0: 8 bytes (+ checksum)<br>:<br>1 1 1 1: 8 bytes (+ checksum) |

#### LSS Bit (Transmission/Reception Continuation Select)

The LSS bit indicates that the data group to be transmitted or received next is not the last data group when response data of 9 bytes or more is to be transmitted or received.

With 0 set, data and checksum are transmitted or received because the next data group to be transmitted or received is the last one.

With 1 set, only data is transmitted or received, and the checksum is not included because the next data group to be transmitted or received is not the last one.

Set the LSS bit only when the FSM bit is 1 (frame separate mode) and response data of 9 bytes or more is to be transmitted or received.

Set the LSS bit only when the RTS bit in the RLN3nLTRC is 0 (response transmit/receive is stopped).

#### FSM Bit (Frame Separate Mode Select)

The FSM bit sets the response communication mode.
With 0 set, frame separate mode is not selected. In this case, after header transmission is started (the FTS bit in the RLN3nLTRC register is 1), response is transmitted/received without the RTS bit in the RLN3nLTRC register being set.
With 1 set, frame separate mode is selected. If the RTS bit of the RLN3nLTRC register is set to 1 during header transmission, response transmission is executed after header transmission is completed.

For response reception or transmission following the header transmission automatically (frame combined mode), set the FSM bit to 0.
When causing a transition to LIN self-test mode, set this bit to 0 before transition.
For details on frame separate mode, see **Section 18.5.3.4 (2), Frame Separate Mode**.
Set this bit when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is halted).
When response data of 9 bytes or more is to be transmitted or received, set the FSM bit to 1.

#### CSM Bit (Checksum Select)

The CSM bit sets the checksum mode.
With 0 set, classic checksum mode is selected.
With 1 set, enhanced checksum mode is selected.
When the timeout error is used (the FTERE bit in the RLN3nLEDE register is 1), the specific timeout time depends on the setting of this bit. For details on the bit error, see **Section 18.5.3.7, Error Status**.
Set this bit when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is halted).
When response data of 9 bytes or more is to be transmitted or received, do not change the CSM bit setting after the first data group through the last data group.
During communication of response data of 9 bytes or more, only the last data group (the LSS bit is 0) includes the checksum, and no other groups (the LSS bit is 1) include the checksum.

#### RFT Bit (Response Field Communication Direction Select)

The RFT bits set the direction of the response field/wake-up signal communication.
With 0 set, reception is performed in the response field. In LIN wake-up mode, wake-up reception is performed (low level width of the input signal is counted).
With 1 set, transmission is performed in the response field. In LIN wake-up mode, wake-up transmission is performed.
Set this bit when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is halted).
When response data of 9 bytes or more is to be transmitted or received, do not change the RFT bit setting after the first data group through the last data group.

#### RFDL[3:0] Bits (Response Field Length Select)

The RFDL bits set the length of the response field data.
The data length can be 0 to 8 bytes excluding the checksum size.
To transmit response data with the FSM bit set to 0 (frame combined mode), set the RFDL bits before header transmission (the FTS bit in the RLN3nLTRC register is 0).
To transmit response data with the FSM bit set to 1 (frame separate mode), set the RFDL bits before response transmission (the RTS bit in the RLN3nLTRC register is 0).
To receive response data, set the RFDL bits before header transmission (the FTS bit in the

RLN3nLTRC register is 0).
When response data of 9 bytes or more is to be transmitted or received, set the RFDL bits before data group transmission/reception (RTS bit in the RLN3nLTRC register is 0).
Only the last data group (the LSS bit is 0) includes the checksum, and no other groups (the LSS bit is 1) include the checksum.

### 18.3.2.17 RLN3nLIDB — LIN ID Buffer Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $15_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IDP1 | IDP0 | ID[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.27 RLN3nLIDB register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | IDP1 | Parity Setting (P1)<br>Sets the parity bits (P1) to be transmitted in the ID field. |
| 6 | IDP0 | Parity Setting (P0)<br>Sets the parity bits (P0) to be transmitted in the ID field. |
| 5 to 0 | ID[5:0] | ID Setting<br>Sets the 6-bit ID value to be transmitted in the ID field. |

Set the RLN3nLIDB register when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is halted).

In LIN self-test mode, this register operates as described below.

Write the value to be transmitted before communication. After completion of frame transmission/reception (after loopback), the reversed value of the received value can be read.

For details about the LIN self-test mode, see **Section 18.5.5, LIN Self-Test Mode**.

#### IDP[1:0] Bits (Parity Setting)

The IDP bits set the parity bits (P0 and P1) to be transmitted in the ID field of the LIN frame. IDP0 for P0 and IDP1 for P1.

Since parity is not automatically calculated, set the calculation result. Note that if the erroneous result is set, it is transmitted as is.

#### ID[5:0] Bits (ID Setting)

The ID bit sets the 6-bit ID value to be transmitted in the ID field of the LIN frame.

### 18.3.2.18 RLN3nLCBR — LIN Checksum Buffer Register (n = 0 to 3)

**Access:** This register can be read only in 8-bit units. In LIN self-test mode, this register can be read/written in 8-bit units.

**Address:** <RLIN3n_base> + $16_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CKSM[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.28 RLN3nLCBR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | CKSM[7:0] | Holds the checksum value transmitted or received. |

In LIN mode, this register operates as follows:

- When the RFT bit in the RLN3nLDFC register is 1 (transmission):
  The value transmitted can be read from the register. Read the value after transmission is completed.
  Writing to this register is invalid.
- When the RFT bit in the RLN3nLDFC register is 0 (reception):
  The value received can be read from the register. Read the value after reception is completed.
  Writing to this register is invalid.

In LIN self-test mode, this register operates as follows:

- When the RFT bit in the RLN3nLDFC register is 1 (transmission):
  After completion of the frame transmission (after loopback), the reversed value of the received value can be read.
- When the RFT bit in the RLN3nLDFC register is 0 (reception):
  Write the value to be received before communication. After completion of frame reception (after loopback), the reversed value of the received value can be read.

For details about the LIN self-test mode, see **Section 18.5.5, LIN Self-Test Mode**.

Set the RLN3nLCBR register when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is halted).

When response data of 9 bytes or more is to be transmitted or received, the checksum is appended only to the last data group; this register is not updated for the other data groups.

### 18.3.2.19 RLN3nLDBRb — LIN Data Buffer b Register (n = 0 to 3, b = 1 to 8)

**Access:** This register can be read/written in 8-bit units.

**Address:** RLN3nLDBR1: <RLIN3n_base> + $18_H$
RLN3nLDBR2: <RLIN3n_base> + $19_H$
RLN3nLDBR3: <RLIN3n_base> + $1A_H$
RLN3nLDBR4: <RLIN3n_base> + $1B_H$
RLN3nLDBR5: <RLIN3n_base> + $1C_H$
RLN3nLDBR6: <RLIN3n_base> + $1D_H$
RLN3nLDBR7: <RLIN3n_base> + $1E_H$
RLN3nLDBR8: <RLIN3n_base> + $1F_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LDB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.29 RLN3nLDBRb register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LDB[7:0] | Sets the data to be transmitted or allows the received data to be read.<br>Setting Range: $00_H$ to $FF_H$ |

- For response transmission:
  The LDBRb registers set the data to be transmitted in the response field.
  Use these registers with the following settings.
  - RFT in RLN3nLDFC register is 1 (transmission)
  - FSM in RLN3nLDFC register is 0 (frame combined mode)
  - FTS bit in RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is halted)

  or
  - RFT in RLN3nLDFC register is 1 (transmission)
  - FSM in RLN3nLDFC register is 1 (frame separate mode)
  - RTS in RLN3nLTRC register is 0 (response transmission/reception is halted)
- For response reception:
  The LDBRb registers hold the data received in the response field.
  The received data is overwritten. If an error is detected, the data up to the byte in which the error was detected are stored in the register.
  Do not read these registers when the FTS bit is 1 (frame transmission or wake-up transmission/reception is started)
- For transmission of response data of 9 bytes or more:
  Use the LDBRb registers with the following settings.
  - RFT in RLN3nLDFC register is 1 (transmission)
  - FSM in RLN3nLDFC register is 1 (frame separate mode)
  - RTS in RLN3nLTRC register is 0 (response transmission/reception is halted)
- For reception of response data of 9 bytes or more:
  Do not read these registers when the RTS bit is 1 (response transmission/reception is started).

- In LIN self-test mode, these registers operate as described below.

Write the value to be transmitted before communication. After completion of frame transmission/ reception (after loopback), the reversed value of the received value can be read.

For details about the LIN self-test mode, see **Section 18.5.5, LIN Self-Test Mode**.

## 18.3.3 LIN Slave Related Registers

### 18.3.3.1 RLN3nLWBR — LIN Wake-up Baud Rate Select Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $01_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NSPB[3:0] | | | | LPRS[2:0] | | | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R |

**Table 18.30 RLN3nLWBR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | NSPB[3:0] | Bit Sampling Count Select<br>b7 b4<br>0 0 0 0: 16 sampling<br>0 0 1 1: 4 sampling<br>0 1 1 1: 8 sampling<br>Settings other than the above are prohibited. |
| 3 to 1 | LPRS[2:0] | Prescaler Clock Select<br>b3 b1<br>0 0 0: 1/1<br>0 0 1: 1/2<br>0 1 0: 1/4<br>0 1 1: 1/8<br>1 0 0: 1/16<br>1 0 1: 1/32<br>1 1 0: 1/64<br>1 1 1: 1/128 |
| 0 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |

Set the RLN3nLWBR register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### NSPB[3:0] Bits (Bit Sampling Count Select)

The NSPB bits select the number of sampling in one Tbit (reciprocal of the baud rate).
When the frame communication is performed in LIN slave mode (fixed baud rate) (LMD[1:0] bits in the RLN3nLMD register = $11_B$), set these bits to "$0000_B$" (16 sampling).
When the frame communication is performed in LIN slave mode (auto baud rate) (LMD[1:0] bits in the RLN3nLMD register = $10_B$), set these bits to "$0011_B$" (4 sampling) or "$0111_B$" (8 sampling).

#### LPRS[2:0] Bits (Prescaler Clock Select)

The LPRS bits select the frequency division ratio for the prescaler. The LIN communication clock source is divided by this prescaler.

The LIN communication clock source is divided by this prescaler.

In LIN slave mode (auto baud rate) (LMD[1:0] bits in the RLN3nLMD register = $10_B$), set these bits so that the prescaler clock becomes as follows according to the target baud rate.

| [Target baud rate] | [Prescaler clock] |
|---|---|
| 1 to 20 kbps | : 4 MHz*1 |
| 1 to 2.4 kbps (excluding 2.4 kbps) | : 4 MHz |
| 2.4 to 20 kbps | : 8 to 12 MHz |

**Note 1.** Use the clock with NSPB bits set to "$0011_B$" (four samplings).

### 18.3.3.2 RLN3nLBRP01 — LIN Baud Rate Prescaler 01 Register (n = 0 to 3)

**Access:** RLN3nLBRP01 can be read/written in 16-bit units.
RLN3nLBRP0 can be read/written in 8-bit units.
RLN3nLBRP1 can be read/written in 8-bit units.

**Address:** RLN3nLBRP01: <RLIN3n_base> + $02_H$
RLN3nLBRP0: <RLIN3n_base> + $02_H$
RLN3nLBRP1: <RLIN3n_base> + $03_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BRP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.31 RLN3nLBRP01 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | BRP[15:0] | Assuming that the value set in this register is L (0 to 65535), the baud rate prescaler divides the frequency of the prescaler clock by L + 1.<br>Setting Range: $0000_H$ to $FFFF_H$ |

Set the RLN3nLBRP01 register when the OMM0 bit in the RLN3nLMST register is 0B (in LIN reset mode).
Assuming that the value set in this register is L, the baud rate prescaler divides the frequency of the clock that is selected by the LPRS bits (prescaler clock select bits) in the RLN3nLWBR register by L + 1.
The RLN3nLBRP01 register can be accessed in 8-bit units by registers RLN3nLBRP0 and RLN3nLBRP1.

### 18.3.3.3 RLN3nLSTC — LIN Self-Test Control Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $04_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | LSTM |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 18.32 RLN3nLSTC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | LIN Self-Test Mode Entry<br>The test mode key values for configuring the RLIN3 module in Self-Test mode. |
| 0 | LSTM | LIN Self-Test Mode<br>0: LIN self test mode is not set.<br>1: LIN self test mode is set. |

The RLN3nLSTC register cancels protection of LIN self-test mode.

Set the RLN3nLSTC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

Writing $A7_H$, $58_H$, and $01_H$ successively to the RLN3nLSTC register places the module into LIN self-test mode.

When successive writing is completed thus placing LIN self-test mode to be entered, the LSTM bit is set to 1.

Do not write any other value during successive writing.

For making transition to LIN self-test mode, see **Section 18.5.5, LIN Self-Test Mode**.

When read, bits 6 to 1 return "$000000_B$", and bit 7 returns an undefined value.

#### LSTM Bit (LIN Self-Test Mode)

When transition to LIN self-test mode is completed, the LSTM bit is set to 1.
For leaving LIN self-test mode, see **Section 18.5.5, LIN Self-Test Mode**.
Writing 1 to this bit does not affect the value of the RLN3nLSTC register if it is not a part of successive writing of $A7_H$, $58_H$, and $01_H$.

### 18.3.3.4 RLN3nLMD — LIN Mode Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $08_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | LRDNFS | LIOS | — | — | LMD[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R | R/W | R/W |

**Table 18.33 RLN3nLMD register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 5 | LRDNFS | LIN Reception Data Noise Filtering Disable<br>0: The noise filter is enabled.<br>1: The noise filter is disabled. |
| 4 | LIOS | LIN Interrupt Output Select<br>0: Not supported, setting is prohibited.<br>1: RLIN3n transmission interrupt, RLIN3n successful reception interrupt, and RLIN3n reception status interrupt are used. |
| 3, 2 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 1, 0 | LMD[1:0] | LIN/UART Mode Select<br>b1 b0<br>1 0: LIN Slave mode with Auto Baud rate<br>1 1: LIN Slave mode with fixed Baud rate |

Set the RLN3nLMD register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### LRDNFS Bit (LIN Reception Data Noise Filtering Disable)

The LRDNFS bit enables or disables the noise filter when receiving data.
With 0 set, the noise filter is enabled when receiving data.
With 1 set, the noise filter is disabled when receiving data.

#### LIOS Bit (LIN Interrupt Output Select)

The LIOS bit selects the number of interrupt outputs from the LIN/UART interface.
With 0 set, the RLIN3n interrupt n is generated from the LIN/UART interface.
With 1 set, the RLIN3n transmission interrupt, RLIN3n successful reception interrupt, and RLIN3n reception status interrupt are generated from the LIN/UART interface.
For each interrupt source, see **Section 18.4, Interrupt Sources**.
The LIOS bit must always be set when using LIN mode operation, in order to enable the interrupt generation by the RLIN3n module. For UART mode, this bit is not relevant.

#### LMD[1:0] Bits (LIN/UART Mode Select)

The LMD bits select the LIN/UART interface mode.
To use this module as an LIN slave, set these bits to "$10_B$" (auto baud rate) or "$11_B$" (fixed baud rate).

### 18.3.3.5 RLN3nLBFC — LIN Break Field Configuration Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $09_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | LBLT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 18.34 RLN3nLBFC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 0 | LBLT | Reception Break (Low-Level) Detection Width Setting<br>0: A break (low-level) is detected in 9.5 or 10 Tbits<br>1: A break (low-level) is detected in 10.5 or 11 Tbits |

Set the RLN3nLBFC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### LBLT Bit (Reception Break (Low-Level) Detection Width Setting)

- When RLN3nLMD.LMD is "$10_B$" (in LIN slave mode (auto baud rate))
  - 0: Low-level width of 10 Tbits or longer is detected.
  - 1: Low-level width of 11 Tbits or longer is detected.
- When RLN3nLMD.LMD is "$11_B$" (in LIN slave mode (fixed baud rate))
  - 0: Low-level width of 9.5 Tbits or longer is detected.
  - 1: Low-level width of 10.5 Tbits or longer is detected.

### 18.3.3.6 RLN3nLSC — LIN Space Configuration Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $0A_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | IBS[1:0] | | — | IBHS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R/W | R/W | R/W |

**Table 18.35 RLN3nLSC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 5, 4 | IBS[1:0] | Inter-Byte Space Select<br>b5 b4<br>0 0: 0 Tbit<br>0 1: 1 Tbit<br>1 0: 2 Tbits<br>1 1: 3 Tbits |
| 3 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 2 to 0 | IBHS[2:0] | Inter-Byte Space (Header)/Response Space Select<br>b2 b0<br>0 0 0: 0 Tbit<br>0 0 1: 1 Tbit<br>0 1 0: 2 Tbits<br>0 1 1: 3 Tbits<br>1 0 0: 4 Tbits<br>1 0 1: 5 Tbits<br>1 1 0: 6 Tbits<br>1 1 1: 7 Tbits |

Set the RLN3nLSC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

This register is enabled only during response transmission, and disabled during response reception.

Some combinations of the set values result in the length of a frame or a response exceeding the timeout time. Set the appropriate values in this register.

#### IBS[1:0] Bits (Inter-Byte Space Select)

These bits set the width of the inter-byte space of the response transmission.
0 Tbit to 3 Tbits can be set.

#### IBHS[2:0] Bits (Inter-Byte Space (Header)/Response Space Select)

The IBHS bits set the width of the response space before response transmission.
0 Tbit to 7 Tbits can be set.

### 18.3.3.7 RLN3nLWUP — LIN Wake-up Configuration Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $0B_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WUTL[3:0] | | | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R |

**Table 18.36 RLN3nLWUP register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | WUTL[3:0] | Wake-up Transmission Low level Width Select<br>b7 b4<br>0 0 0 0: 1 Tbit<br>0 0 0 1: 2 Tbits<br>0 0 1 0: 3 Tbits<br>0 0 1 1: 4 Tbits<br>:<br>1 1 0 0: 13 Tbits<br>1 1 0 1: 14 Tbits<br>1 1 1 0: 15 Tbits<br>1 1 1 1: 16 Tbits |
| 3 to 0 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |

Set the RLN3nLWUP register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### WUTL[3:0] Bits (Wake-up Transmission Low Level Width Select)

These bits set the low-level width of the wake-up frame transmission.
1 Tbit to 16 Tbits can be set.

### 18.3.3.8 RLN3nLIE — LIN Interrupt Enable Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $0C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SHIE | ERRIE | FRCIE | FTCIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 18.37 RLN3nLIE register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 3 | SHIE | Successful Header Transmission Interrupt Enable<br>0: Disables successful header transmission interrupt.<br>1: Enables successful header transmission interrupt. |
| 2 | ERRIE | Error Detection Interrupt Enable<br>0: Disables error detection interrupt.<br>1: Enables error detection interrupt. |
| 1 | FRCIE | Successful Response/Wake-up Reception Interrupt Enable<br>0: Disables successful Response/wake-up reception interrupt.<br>1: Enables successful Response/wake-up reception interrupt. |
| 0 | FTCIE | Successful Response/Wake-up Transmission Interrupt Enable<br>0: Disables successful Response/wake-up transmission interrupt.<br>1: Enables successful Response/wake-up transmission interrupt. |

Set the RLN3nLIE register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### SHIE Bit (Successful Header Reception Interrupt Enable)

The SHIE bit enables or disables interrupt generation upon successful reception of a header.
With 0 set, the interrupt is not generated when the HTRC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt is generated when the HTRC flag in the RLN3nLST register is set to 1.

#### ERRIE Bit (Error Detection Interrupt Enable)

The ERRIE bit enables or disables interrupt generation upon detection of an error.
With 0 set, the interrupt is not generated when the ERR flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt is generated when the ERR flag in the RLN3nLST register is set to 1.
Interrupt sources can be the bit error, frame/response timeout error, framing error, sync filed error, ID parity error, checksum error, and response preparation error.
Detection of the bit error, frame/response timeout error, sync filed error, ID parity error, and framing error can be enabled or disabled using the RLN3nLEDE register.

### FRCIE Bit (Successful Response/Wake-up Reception Interrupt Enable)

The FRCIE bit enables or disables interrupt generation upon successful reception of a response or a wake-up frame (counting of low level width of the input signal).
With 0 set, the interrupt is not generated when the FRC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt is generated when the FRC flag in the RLN3nLST register is set to 1.

### FTCIE Bit (Successful Response/Wake-up Transmission Interrupt Enable)

The FTCIE bit enables or disables interrupt generation upon successful transmission of a response or a wake-up frame.
With 0 set, the interrupt is not generated when the FTC flag in the RLN3nLST register is set to 1.
With 1 set, the interrupt is generated when the FTC flag in the RLN3nLST register is set to 1.

### 18.3.3.9 RLN3nLEDE — LIN Error Detection Enable Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $0D_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LTES | IPERE | — | SFERE | FERE | TERE | — | BERE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W |

**Table 18.38 RLN3nLEDE register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | LTES | Timeout Error Select<br>0: Frame timeout error<br>1: Response timeout error |
| 6 | IPERE | ID Parity Error Detection Enable<br>0: Disables ID Parity error detection.<br>1: Enables ID Parity error detection. |
| 5 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 4 | SFERE | Sync Field Error Detection Enable<br>0: Disables Sync Field error detection.<br>1: Enables Sync Field error detection. |
| 3 | FERE | Framing Error Detection Enable<br>0: Disables framing error detection.<br>1: Enables framing error detection. |
| 2 | TERE | Timeout Error Detection Enable<br>0: Disables frame/response timeout error detection.<br>1: Enables frame/response timeout error detection. |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 0 | BERE | Bit Error Detection Enable<br>0: Disables bit error detection.<br>1: Enables bit error detection. |

Set the RLN3nLEDE register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### LTES Bit (Timeout Error Select)

The LTES bit selects the timeout function to be used.
With 0 set, the timeout function applies to frame timeout.
With 1 set, the timeout function applies to response timeout.
For details on the timeout error, see **Section 18.5.3.7, Error Status**.

#### IPERE Bit (ID Parity Error Detection Enable)

This bit enables or disables detection of the ID parity error.
With 0 set, the ID parity error is not detected.
With 1 set, the ID parity error is detected.
When this bit is set to 1, the detection result is reflected in the IPER flag in the RLN3nLEST register.
For details on the ID parity error, see **Section 18.5.3.7, Error Status**.

#### SFERE Bit (Sync Field Error Detection Enable)

This bit enables or disables detection of the sync field error.
With 0 set, the sync field error is not detected.
With 1 set, the sync field error is detected.
Regardless of the setting of this bit, when a sync field error is detected, this module waits for the next header.
When this bit is set to 1, the detection result is reflected in the SFER flag in the RLN3nLEST register.
For details on the sync filed error, see **Section 18.5.3.7, Error Status**.

#### FERE Bit (Framing Error Detection Enable)

The FERE bit enables or disables detection of the framing error.
With 0 set, the framing error is not detected.
With 1 set, the framing error is detected.
When this bit is set to 1, the detection result is indicated in the FER flag in the RLN3nLEST register.
For details on the framing error, see **Section 18.5.3.7, Error Status**.

#### TERE Bit (Timeout Error Detection Enable)

The TERE bit enables or disables detection of the frame timeout error or the response timeout error.
With 0 set, the frame timeout error or response timeout error is not detected.
With 1 set, the frame timeout error or response timeout error is detected.
When this bit is set to 1, the detection result is indicated in the TER flag in the RLN3nLEST register.
With the LTES bit, either the frame timeout error or response timeout error can be selected.
The timeout error should not be used in LIN slave mode [auto baud rate] (when the LMD[1:0] bits in the RLN3nLMD register are "$10_B$").
Do not use the timeout error if response data of 9 bytes or more is to be transmitted or received.
For details on the timeout error, see **Section 18.5.3.7, Error Status**.

#### BERE Bit (Bit Error Detection Enable)

The BERE bit enables or disables detection of the bit error.
With 0 set, the bit error is not detected.
With 1 set, the bit error is detected.
When this bit is set to 1, the detection result is indicated in the BER flag in the RLN3nLEST register.
For details on the bit error, see **Section 18.5.3.7, Error Status**.

### 18.3.3.10 RLN3nLCUC — LIN Control Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $0E_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | OM1 | OM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 18.39 RLN3nLCUC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 1 | OM1 | LIN Mode Select<br>0: LIN wake-up mode is caused.<br>1: LIN operation mode is caused. |
| 0 | OM0 | LIN Reset<br>0: LIN reset mode is caused.<br>1: LIN reset mode is canceled. |

Set the RLN3nLCUC register to $01_H$ to cause a transition to LIN wake-up mode after canceling LIN reset mode, or set the register to $03_H$ to cause a transition to LIN operation mode.

In LIN self-test mode, set the RLN3nLCUC register to $03_H$ after a transition to LIN self-test mode is completed.

After a value is written to this register, confirm that the value written is actually indicated in the RLN3nLMST register before writing another value.

#### OM1 Bit (LIN Mode Select)

The OM1 bit selects the specific LIN operation mode (either LIN wake-up mode or LIN operation mode) after canceling LIN reset mode.
With 0 set, LIN wake-up mode is caused.
With 1 set, LIN operation mode is caused.
This bit is valid only when the OMM0 bit in the RLN3nLMST register is 1.
Writing a value to this bit is disabled while the FTS bit in the RLN3nLTRC register is 1.

#### OM0 Bit (LIN Reset)

The OM0 bit selects either causing a transition to LIN reset mode or canceling LIN reset mode.
With 0 set, LIN reset mode is caused.
With 1 set, LIN reset mode is canceled.

It takes 3 peripheral clock cycles + 4 communication clock cycles, until the OMM0 bit in the RLN3nLMST register reflects the OM0 bit setting in the RLN3nLCUC register.

### 18.3.3.11 RLN3nLTRC — LIN Transmission Control Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $10_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | LNRR | RTS | FTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W |

**Table 18.40 RLN3nLTRC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 2 | LNRR | No LIN Response Request<br>0: Response for the reception ID<br>1: No response for the reception ID |
| 1 | RTS | Response Transmission/Reception Start<br>0: Response transmission/reception is stopped.<br>1: Response transmission/reception is started. |
| 0 | FTS | LIN Communication Start<br>0: Header reception/wake-up transmission/reception is stopped.<br>1: Header reception/wake-up transmission reception is started. |

#### LNRR Bit (No LIN Response Request)

After receiving the header and checking the received ID, set this bit to 1 if no response is transmitted/received.
Once set, this bit is automatically cleared to 0 upon detection of new sync field or transition to LIN reset mode.
Only 1 can be written to this bit; 0 cannot be written.

To write 1 to this bit, write $04_H$ using the single memory storing instruction.
Do not set this bit and the RTS bit to 1 simultaneously.

Writing a value to this bit is disabled when the OMM0 bit of the RLN3nLMST register is $0_B$ (LIN reset mode).

Writing a value to this bit is disabled when the FTS bit is 0 (header reception or wake-up transmission/reception is halted).

#### RTS Bit (Response Transmission/Reception Start)

After receiving the header and checking the received ID, set this bit to 1 at the response transmission or at the start of response reception.
After receiving the header and checking the received ID, set this bit to 1 at the start of response transmission/reception.
Once set, this bit is automatically cleared to 0 upon completion of response or transition to LIN reset mode.
Only 1 can be written to this bit; 0 cannot be written.
To write 1 to this bit, write $02_H$ to the RLN3nLTRC register using the single memory storing instruction.

Do not set this bit and the LNRR bit to 1 simultaneously
Writing a value to this bit is disabled when the OMM0 bit of the RLN3nLMST register is $0_B$ (LIN reset mode).

Writing a value to this bit is disabled when the FTS bit is 0 (header reception or wake-up transmission/reception is halted).
When response data of 9 bytes or more is to be transmitted or received, set this bit to 1 each time a data group (variable from 0 to 8 bytes) is transmitted or received. Once set, this bit is automatically cleared to 0 upon completion of data group transmission/reception or transition to LIN reset mode.

#### FTS Bit (LIN Communication Start)

Set this bit to 1 to start header reception or wake-up transmission/reception.
Also set this bit to 1 to allow wake-up reception (counting of the low level width of the input signal).
Only 1 can be written to this bit; 0 cannot be written.
Writing a value to this bit is disabled when the OMM0 bit of the RLN3nLMST register is $0_B$ (LIN reset mode).

This bit is set to 0 upon completion of frame or wake-up communication and transition to LIN reset mode.

### 18.3.3.12 RLN3nLMST — LIN Mode Status Register (n = 0 to 3)

**Access:** This register can be read only in 8-bit units

**Address:** <RLIN3n_base> +$11_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | OMM1 | OMM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 18.41 RLN3nLMST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>Writing is ignored. |
| 1 | OMM1 | LIN Mode Status Monitor<br>0: The module is in LIN wake-up mode.<br>1: The module is in LIN operation mode. |
| 0 | OMM0 | LIN Reset Status Monitor<br>0: The module is in LIN reset mode.<br>1: The module is not in LIN reset mode. |

#### OMM1 Bit (LIN Mode Status Monitor)

The OMM1 bit indicates the current operating mode.

#### OMM0 Bit (LIN Reset Status Monitor)

The OMM0 bit indicates the current operating mode.

It takes 3 peripheral clock cycles + 4 communication clock cycles, until the OMM0 bit in the RLN3nLMST register reflects the OM0 bit setting in the RLN3nLCUC register.

### 18.3.3.13 RLN3nLST — LIN Status Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $12_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | HTRC | D1RC | — | — | ERR | — | FRC | FTC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R/W | R/W |

**Table 18.42 RLN3nLST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | HTRC | Successful Header Reception Flag<br>0: Header transmission has not been completed.<br>1: Header transmission has been completed. |
| 6 | D1RC | Successful 1st data byte Reception Flag<br>These bits are always read as 0. The write value should always be 0. |
| 5, 4 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 3 | ERR | Error Detection Flag<br>0: No error has been detected.<br>1: Error has been detected. |
| 2 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 1 | FRC | Successful Response/Wake-up Reception Flag<br>0: Response or wake-up reception has not been completed.<br>1: Response or wake-up reception has been completed. |
| 0 | FTC | Successful Response/Wake-up Transmission Flag<br>0: Response or wake-up transmission has not been completed.<br>1: Response or wake-up transmission has been completed. |

The RLN3nLST register is automatically cleared to $00_H$ upon transition to LIN reset mode.

In LIN reset mode, writing a value to this register is disabled. In LIN reset mode, the register retains $00_H$.

To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the single memory storing instruction.

#### HTRC Flag (Successful Header Reception Flag)

Only 0 can be written to the HTRC flag; when 1 is written, the bit retains the value before 1 is written. The HTRC flag is set to 1 upon completion of header reception. Here, an interrupt request is generated if the SHIE bit in the RLN3nLIE register is 1 (interrupt is enabled). However, if header reception is completed while this bit is 1, an interrupt is not generated. To clear this bit to 0, write 0 to the bit.
To detect a new header in the response field upon completion of header reception, clear this bit after it is set to 1.

#### D1RC Flag (Successful 1st data byte Reception Flag)

Only 0 can be written to the D1RC flag; when 1 is written, the bit retains the value that has been retained before 1 is written.

The D1RC flag is set to 1 upon completion of 1st data byte reception. Here, an interrupt request is not generated. Write 0 to clear this bit.
When response data of 9 bytes or more is to be received, this bit is set to 1 each time 1st data byte of a data group (variable from 0 to 8 bytes) is received. Write 0 before starting reception of the next data group.

### ERR Flag (Error Detection Flag)

The ERR flag is set to 1 upon detection of an error (when the value of at least one of the flags of the RLN3nLEST register is set to 1). Here, an interrupt is generated if the ERRIE bit in the RLN3nLIE register is 1 (interrupt is enabled). However, if an error is detected while this bit is 1, an interrupt is not generated. To clear the bit to 0, write 0 to the RPER, IPER, CSER, SFER, FER, TER, and BER flags in the RLN3nLEST register.

### FRC Flag (Successful Response/Wake-up Reception Flag)

Only 0 can be written to the FRC flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The FRC flag is set to 1 upon completion of response or wake-up reception. Here, an interrupt is generated if the FRCIE bit in the RLN3nLIE register is 1 (interrupt is enabled). However, if response reception or wake-up reception is completed while this bit is 1, an interrupt is not generated. Write 0 to clear this bit.
When response data of 9 bytes or more is to be received, this bit is set to 1 each time a data group (variable from 0 to 8 bytes) is received. Write 0 before starting reception of the next data group.

### FTC Flag (Successful Response/Wake-up Transmission Flag)

Only 0 can be written to the FTC flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The FTC flag is set to 1 upon completion of response or wake-up transmission. Here, an interrupt is generated if the FTCIE bit in the RLN3nLIE register is 1 (interrupt is enabled). However, if response reception or wake-up reception is completed while this bit is 1, an interrupt is not generated. Write 0 to clear this bit.
When response data of 9 bytes or more is to be transmitted, this bit is set to 1 each time a data group (variable from 0 to 8 bytes) is transmitted. Write 0 before starting transmission of the next data group.

### 18.3.3.14 RLN3nLEST — LIN Error Status Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $13_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RPER | IPER | CSER | SFER | FER | TER | — | BER |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W |

**Table 18.43 RLN3nLEST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RPER | Response Preparation Error Flag<br>0: Response preparation error has not been detected.<br>1: Response preparation error has been detected. |
| 6 | IPER | ID Parity Error Flag<br>0: ID parity error has not been detected.<br>1: ID parity error has been detected. |
| 5 | CSER | Checksum Error Flag<br>0: Checksum error has not been detected.<br>1: Checksum error has been detected. |
| 4 | SFER | Sync Field Error Flag<br>0: Sync field error has not been detected.<br>1: Sync field error has been detected. |
| 3 | FER | Framing Error Flag<br>0: Framing error has not been detected.<br>1: Framing error has been detected. |
| 2 | TER | Timeout Error Flag<br>0: Frame/response timeout error has not been detected.<br>1: Frame/response timeout error has been detected. |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 0 | BER | Bit Error Flag<br>0: Bit error has not been detected.<br>1: Bit error has been detected. |

The RLN3nLEST register is automatically cleared to $00_H$ upon transition to LIN reset mode.

In LIN reset mode, this register cannot be written to. In LIN reset mode, the register retains $00_H$.

To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the single memory storing instruction.

#### RPER Flag (Response Preparation Error Flag)

Only 0 can be written to the RPER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The RPER flag is set to 1 when response preparation is not completed before reception of the first byte of response is completed. Write 0 to clear this bit.

#### IPER Flag (ID Parity Error Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value that has been retained before 1 is written.

This bit is set to 1 when the received identifier parity bits do not match the calculated identifier parity bit values and the corresponding error detection is enabled. Write 0 to clear this bit.

#### CSER Flag (Checksum Error Flag)

Only 0 can be written to the CSER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The CSER flag is set to 1 when the received checksum value during response reception does not match the internally calculated checksum value. Write 0 to clear this bit.

#### SFER Flag (Sync Field Error Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
This bit is set to 1 if the sync field is not detected as "$55_H$" and the break low width is more than or equal to the configured break low width. Write 0 to clear this bit.

#### FER Flag (Framing Error Flag)

Only 0 can be written to the FER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The FER flag is set to 1 when a '0' value is sampled as STOP bit during reception and the corresponding error detection is enabled. Write 0 to clear this bit.

#### TER Flag (Timeout Error Flag)

Only 0 can be written to the TER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
This flag is set to 1 when the internal timeout counter (frame or response timeout) reaches the error threshold value (calculated automatically) and the corresponding error detection is enabled. Write 0 to clear this bit.

#### BER Flag (Bit Error Flag)

Only 0 can be written to the BER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The BER flag is set to 1 when the sampled bit value does not match the transmitted bit value during transmission and the corresponding error detection is enabled. Write 0 to clear this bit.

### 18.3.3.15 RLN3nLDFC — LIN Data Field Configuration Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units.

**Address:** <RLIN3n_base> + $14_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LSS | — | LCS | RCDS | RFDL[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.44 RLN3nLDFC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | LSS | Transmission/Reception Continuation Select<br>0: The data group to be transmitted/received next is the last one.<br>1: The data group to be transmitted/received next is not the last one. (Checksum is not included.) |
| 6 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 5 | LCS | Checksum Select<br>0: Classic checksum mode<br>1: Enhanced checksum mode |
| 4 | RCDS | Response Field Communication Direction Select<br>0: Reception<br>1: Transmission |
| 3 to 0 | RFDL[3:0] | Response Field Length Select<br>b3 b0<br>0 0 0 0: 0 byte (+ checksum)<br>0 0 0 1: 1 byte (+ checksum)<br>0 0 1 0: 2 bytes (+ checksum)<br>:<br>0 1 1 1: 7 bytes (+ checksum)<br>1 0 0 0: 8 bytes (+ checksum)<br>1 0 0 1: 8 bytes (+ checksum)<br>1 0 1 0: 8 bytes (+ checksum)<br>:<br>1 1 1 1: 8 bytes (+ checksum) |

#### LSS Bit (Transmission/Reception Continuation Select)

The LSS bit indicates that the data group to be transmitted or received next is not the last data group when response data of 9 bytes or more is to be transmitted or received. With 0 set, data and checksum are transmitted or received because the next data group to be transmitted or received is the last one. With 1 set, only data is transmitted or received, and the checksum is not included because the next data group to be transmitted or received is not the last one.
During LIN communication, do not set “1” to this bit.
This should be set when the RTS bit is 0 (response transmission/reception stopped).

#### LCS Bit (Checksum Select)

The LCS bit sets the checksum mode.
With 0 set, classic checksum mode is selected.
With 1 set, enhanced checksum mode is selected.
When the timeout error is used (the TERE bit in the RLN3nLEDE register is 1), the specific timeout

time depends on the setting of this bit. For details on the bit error, see **Section 18.5.3.7, Error Status**.
When the length of the response field data is 0 byte (the RFDL bit is 0), do not set this bit to “1” (enhanced).
When response data of 9 bytes or more is to be transmitted or received, do not change the LCS bit setting after the first data group through the last data group.
During transmission or reception of response data of 9 bytes or more, only the last data group (the LSS bit is 0) includes the checksum, and no other groups (the LSS bit is 1) include the checksum.
When response data of 9 bytes or more is to be transmitted or received, set the FSM bit to 1.
This should be set when the RTS bit is 0 (response transmission/reception stopped).

#### RCDS Bit (Response Field Communication Direction Select)

This bit selects the direction of the response field/wake-up signal communication.
With 0 set, reception is performed in the response field. In LIN wake-up mode, wake-up reception is performed (low-level width of the input signal is counted).
With 1 set, transmission is performed in the response field. In LIN wake-up mode, wake-up transmission is performed.
This bit should be set when the RTS bit in the RLN3nLTRC register is 0 in LIN operation mode (response transmission/reception stopped) or when the FTS bit is 0 in LIN wake-up mode (header reception or wake-up transmission/reception stopped).
When response data of 9 bytes or more is to be transmitted or received, do not change this bit setting after the first data group through the last data group.

#### RFDL[3:0] Bits (Response Field Length Select)

The RFDL bits set the length of the response field data.
The data length can be 0 to 8 bytes excluding the checksum size.
These bits should be set when the RTS bit in the RLN3nLTRC register is 0 (response transmission/reception stopped).
When response data of 9 bytes or more is to be transmitted, only the last data group (the LSS bit is 0) includes the checksum, and no other groups (the LSS bit in the RLN3nLDFC register is 1) include the checksum.

### 18.3.3.16 RLN3nLIDB — LIN/UART ID Buffer Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $15_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IDP1 | IDP0 | ID[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.45 RLN3nLIDB register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | IDP[1:0] | Parity Setting<br>Stores the parity bits (P0 and P1) to be received in the ID field. |
| 5 to 0 | ID[5:0] | ID Setting<br>Stores the 6-bit ID value to be received in the ID field. |

The value in the RLN3nLIDB register is enabled after the completion of header reception. In LIN mode (LIN operation mode, LIN wake-up mode), writing to this register is disabled.

In LIN self-test mode, the operation is as follows.

Write the value to be transmitted before communication. After completion of frame transmission/ reception (after loopback), the reversed value of the received value can be read. For details about the LIN self-test mode, see **Section 18.5.5, LIN Self-Test Mode**.

#### IDP[1:0] Bits (Parity Setting)

The IDP bits store the parity bits (P0 and P1) to be received in the ID field of the LIN frame. IDP0 is for P0 and IDP1 is for P1.

When the IPERE bit in the RLN3nLEDE register is 1 (ID parity detection enable), the received value and the value calculated internally are checked. If they do not match, IPER (ID parity error flag) is set.

#### ID[5:0] Bits (ID Setting)

The ID bits store the 6-bit ID value to be received in the ID field of the LIN frame.

### 18.3.3.17 RLN3nLCBR — LIN Checksum Buffer Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units. However, in LIN self-test mode, this register can be read and written in 8-bit units.

**Address:** <RLIN3n_base> + $16_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CKSM[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.46 RLN3nLCBR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | CKSM[7:0] | Holds the checksum value transmitted or received. |

In LIN mode, this register operates as follows:

- When the RCDS bit in the RLN3nLDFC register is 1 (transmission):
  The value transmitted can be read from the register. Read the value after transmission is completed.
  Writing to this register is invalid.
- When the RCDS bit in the RLN3nLDFC register is 0 (reception):
  The value received can be read from the register. Read the value after reception is completed.
  Writing to this register is invalid.

When response data of 9 bytes or more is to be transmitted or received, the checksum is appended only to the last data group; this register is not updated for the other data groups.

In LIN self-test mode, this register operates as follows:

- When the RCDS bit in the RLN3nLDFC register is 1 (transmission):
  After completion of the frame transmission (after loopback), the reversed value of the received value can be read.
- When the RCDS bit in the RLN3nLDFC register is 0 (reception):
  Write the value to be received before communication. After completion of frame transmission/reception (after loopback), the reversed value of the received value can be read.

For details about the LIN self-test mode, see **Section 18.5.5, LIN Self-Test Mode**.

Set the RLN3nLCBR register when the FTS bit in the RLN3nLTRC register is 0 (frame transmission or wake-up transmission/reception is halted).

### 18.3.3.18 RLN3nLDBRb — LIN Data Buffer b Register (n = 0 to 3, b = 1 to 8)

**Access:** This register can be read/written in 8-bit units.

**Address:** RLN3nLDBR1: <RLIN3n_base> + $18_H$
RLN3nLDBR2: <RLIN3n_base> + $19_H$
RLN3nLDBR3: <RLIN3n_base> + $1A_H$
RLN3nLDBR4: <RLIN3n_base> + $1B_H$
RLN3nLDBR5: <RLIN3n_base> + $1C_H$
RLN3nLDBR6: <RLIN3n_base> + $1D_H$
RLN3nLDBR7: <RLIN3n_base> + $1E_H$
RLN3nLDBR8: <RLIN3n_base> + $1F_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LDB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.47 RLN3nLDBRb register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LDB[7:0] | Sets the data to be transmitted or allows the received data to be read.<br>Setting Range: $00_H$ to $FF_H$ |

- For response transmission:
  The RLN3nLDBRb registers set the data to be transmitted in the response field.
  These registers should be set when the RTS bit in the RLN3nLTRC register is 0 (response transmission/reception is halted).
- For response reception:
  The RLN3nLDBRb registers hold the data received in the response field.
  The received data is overwritten. If an error is detected, the data up to the byte in which the error was detected are stored in the register.
  Do not read these registers when the RTS bit is 1 (response transmission/reception is started)

In LIN self-test mode, the operation is as follows.

Write the value to be transmitted before communication. After completion of frame transmission/reception (after loopback), the reversed value of the received value can be read. For details about the LIN self-test mode, see **Section 18.5.5, LIN Self-Test Mode**.

## 18.3.4 UART Related Registers

### 18.3.4.1 RLN3nLWBR — LIN Wake-up Baud Rate Select Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $01_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NSPB[3:0] | | | | LPRS[2:0] | | | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R |

**Table 18.48 RLN3nLWBR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | NSPB[3:0] | Bit Sampling Count Select<br>b7 b4<br>0 1 0 1: 6 sampling<br>0 1 1 0: 7 sampling<br>0 1 1 1: 8 sampling<br>1 0 0 0: 9 sampling<br>1 0 0 1: 10 sampling<br>1 0 1 0: 11 sampling<br>1 0 1 1: 12 sampling<br>1 1 0 0: 13 sampling<br>1 1 0 1: 14 sampling<br>1 1 1 0: 15 sampling<br>1 1 1 1: 16 sampling<br>Settings other than the above are prohibited. |
| 3 to 1 | LPRS[2:0] | Prescaler Clock Select<br>b3 b1<br>0 0 0: 1/1<br>0 0 1: 1/2<br>0 1 0: 1/4<br>0 1 1: 1/8<br>1 0 0: 1/16<br>1 0 1: 1/32<br>1 1 0: 1/64<br>1 1 1: 1/128 |
| 0 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |

Set the LN3nLWBR register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### NSPB[3:0] Bits (Bit Sampling Count Select)

The NSPB bits select the number of sampling in one Tbit (reciprocal of the bit rate).
In UART mode, it is possible to set the NSPB bits from 6 sampling to 16 sampling.

#### LPRS[2:0] Bits (Prescaler Clock Select)

The LPRS bits select the frequency division ratio for the prescaler.

The LIN communication clock source is divided by this prescaler.

### 18.3.4.2 RLN3nLBRP01 — UART Baud Rate Prescaler 01 Register (n = 0 to 3)

**Access:** RLN3nLBRP01 register can be read/written in 16-bit units.
RLN3nLBRP0 register can be read/written in 8-bit units.
RLN3nLBRP1 register can be read/written in 8-bit units.

**Address:** RLN3nLBRP01: <RLIN3n_base> + $02_H$
RLN3nLBRP0: <RLIN3n_base> + $02_H$
RLN3nLBRP1: <RLIN3n_base> + $03_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BRP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.49 RLN3nLBRP01 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | LBRP0[15:0] | Assuming that the value set in this register is L (0 to 65535), the baud rate prescaler divides the frequency of the prescaler clock by L + 1.<br>Setting Range: $0000_H$ to $FFFF_H$ |

Assuming that the value set in this register is L, the baud rate prescaler divides the frequency of the clock that is selected by the LPRS bits (prescaler clock select bits) in the RLN3nLWBR register by L + 1.

The RLN3nLBRP01 register can be accessed in 8-bit units by the registers RLN3nLBRP0 and RLN3nLBRP1

### 18.3.4.3 RLN3nLMD — UART Mode Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $08_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | LRDNFS | — | — | — | LMD[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R | R | R | R/W | R/W |

**Table 18.50 RLN3nLMD register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 5 | LRDNFS | LIN Reception Data Noise Filtering Disable<br>0: The noise filter is enabled.<br>1: The noise filter is disabled. |
| 4 to 2 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 1, 0 | LMD[1:0] | LIN/UART Mode Select<br>b1 b0<br>0 1: UART mode |

Set the RLN3nLMD register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### LRDNFS Bit (UART Reception Data Noise Filtering Disable)

The LRDNFS bit enables or disables the noise filter when receiving data.
With 0 set, the noise filter is enabled when receiving data.
With 1 set, the noise filter is disabled when receiving data.

#### LMD[1:0] Bits (LIN/UART Mode Select)

The LMD bits select the LIN/UART interface mode.
To use the LIN/UART interface as an LIN master, set these bits to $01_B$.

### 18.3.4.4 RLN3nLBFC — UART Configuration Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $09_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | UTPS | URPS | UPS[1:0] | | USBLS | UBOS | UBLS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.51 RLN3nLBFC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 6 | UTPS | UART Output Polarity Switch<br>0: Transmit data normal output<br>1: Transmit data with inversion output |
| 5 | URPS | UART Input Polarity Switch<br>0: Reception data normal output<br>1: Reception data with inversion output |
| 4, 3 | UPS[1:0] | UART Parity Select<br>00: Parity prohibited<br>01: Even Parity<br>10: 0 Parity<br>11: Odd parity |
| 2 | USBLS | UART Stop Bit length Select<br>0: Stop bit:1 bit<br>1: Stop bit: 2 bits |
| 1 | UBOS | UART Transfer Format Order Select<br>0: LSB First<br>1: MSB First |
| 0 | UBLS | UART Character Length Select<br>0: UART 8 bits communication<br>1: UART 7 bits communication |

Set the RLN3nLBFC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### UTPS Bit (UART Output Polarity Switch)

Sets the output polarity for UART communication.
With 0 set, transmit data is output without inversion.
With 1 set, receive data is output with inversion.
The setting of this bit is valid in all the bits of the UART frame.
In half-duplex communication using a single data line, this setting should match with the setting of URPS bit.

#### URPS Bit (UART Input Polarity Switch)

Sets the input polarity for UART communication.
With 0 set, receive data is input without inversion.
With 1 set, receive data is input with inversion.

The setting of this bit is valid in all the bits of the UART frame.
In half-duplex communication using a single data line, this setting should match with the setting of UTPS bit.

### UPS[1:0] Bits (UART Parity Select)

Sets the UART parity.

- When these bits are set to "$00_B$", data is communicated without the parity.

  [Transmission]

  A parity bit is not added to transmit data.

  [Reception]

  Data is received without parity processing. A parity error does not occur, because the parity bit must not exist in the UART frame.

- When these bits are set to "01", data is communicated with the even parity.

  [Transmission]

  If the number of 1s in transmit data is odd, "1" is added to the parity bit. If the number of 1s in transmit data is even, "0" is added to the parity bit.

  [Reception]

  If the number of 1s in receive data including the parity bit is odd, a parity error occurs.

- When these bits are set to "10", data is communicated with 0 parity.

  [Transmission]

  Regardless of the number 1s in transmit data, "0" is added to the parity bit.

  [Reception]

  The value of the parity bit is not judged. Therefore, no parity error occurs.

- When these bits are set to "11", data is communicated with the odd parity.

  [Transmission]

  If the number of 1s in transmit data is odd, "0" is added to the parity bit. If the number of 1s in transmit data is even, "1" is added to the parity bit.

  [Reception]

  If the number of 1s in receive data including the parity bit is even, a parity error occurs.

### USBLS Bit (UART Stop Bit Length Select)

Sets the stop bit length of data for UART communication.
With 0 set, stop bit length of 1 bit is selected.
With 1 set, stop bit length of 2 bits is selected.

### UBOS Bit (UART Transfer Format Select)

Sets the bit order of data for UART communication.
With 0 set, LSB first is selected.
With 1 set, MSB first is selected.

### UBLS Bit (UART Character Length Select)

Sets the character length of one frame for UART communication.
With 0 set, the character length is 8 bits.
With 1 set, the character length is 7 bits.
When the character length of one frame is 9 bits (the UEBE bit in the RLN3nLUOR1 register is 1), the setting of this bit is ignored.

### 18.3.4.5 RLN3nLSC — UART Space Configuration Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $0A_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | IBS[1:0] | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R | R | R |

**Table 18.52 RLN3nLSC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 5, 4 | IBS[1:0] | Inter-Byte Space Select<br>b5 b4<br>0 0: 0 Tbit<br>0 1: 1 Tbit<br>1 0: 2 Tbits<br>1 1: 3 Tbits |
| 3 to 0 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |

Set the RLN3nLSC register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in LIN reset mode).

#### IBS[1:0] Bits (Inter-Byte Space Select)

The IBS bits set the width of the inter-byte space during multi-byte UART transmission.

0 Tbit to 3 Tbits can be set.

### 18.3.4.6 RLN3nLEDE — UART Error Detection Enable Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $0D_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | FERE | OERE | — | BERE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R | R/W |

**Table 18.53 RLN3nLEDE register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 3 | FERE | Framing Error Detection Enable<br>0: Disables framing error detection.<br>1: Enables framing error detection. |
| 2 | OERE | Overrun Error Detection Enable<br>0: Disables overrun error detection.<br>1: Enables overrun error detection. |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 0 | BERE | Bit Error Detection Enable<br>0: Disables bit error detection.<br>1: Enables bit error detection. |

Set the RLN3nLEDE register when the OMM0 bit in the RLN3nLMST register is $0_B$ (in Software reset mode).

#### FERE Bit (Framing Error Detection Enable)

The FERE bit enables or disables detection of the framing error.
With 0 set, the framing error is not detected.
With 1 set, the framing error is detected.
When this bit is set to 1, the detection result is indicated in the FER flag in the RLN3nLEST register.
For details on the framing error, see **Section 18.5.4.5, Error Status**.

#### OERE Bit (Overrun Error Detection Enable)

This bit enables or disables detection of the overrun error.
With 0 set, the overrun error is not detected.
With 1 set, the overrun error is detected.
When this bit is set to 1, the detection result is reflected in the OER flag in the RLN3nLEST register.
For details on the overrun error, see **Section 18.5.4.5, Error Status**.

#### BERE Bit (Bit Error Detection Enable)

The BERE bit enables or disables detection of the bit error.
With 0 set, the bit error is not detected.
With 1 set, the bit error is detected.
When this bit is set to 1, the detection result is indicated in the BER flag in the RLN3nLEST register.

In full-duplex communication, do not set this bit to “1”.
Do not set this register when the NSPB bits in the RLN3nLWBR register is $0101_B$ (6 sampling) and the LRDNFS bit in the RLN3nLMD register is 0 (noise filter is enabled).

For details on the bit error, see **Section 18.5.4.5, Error Status**.

### 18.3.4.7 RLN3nLCUC — UART Control Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units
**Address:** <RLIN3n_base> + $0E_H$
**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | OM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 18.54 RLN3nLCUC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 0 | OM0 | Software Reset<br>0: Software reset mode is caused.<br>1: Software reset mode is canceled. |

After a value is written to this register, confirm that the value written is actually indicated in the RLN3nLMST register before writing another value.

#### OM0 Bit (Software Reset)

The OM0 bit selects either causing a transition to reset mode or canceling reset mode.
With 0 set, reset mode is caused.
With 1 set, reset mode is canceled.

It takes 3 peripheral clock cycles + 4 communication clock cycles, until the OMM0 bit in the RLN3nLMST register reflects the OM0 bit setting in the RLN3nLCUC register.

### 18.3.4.8 RLN3nLTRC — UART Transmission Control Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $10_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RTS | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R |

**Table 18.55 RLN3nLTRC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 1 | RTS | UART Buffer Transmission Start<br>0: UART Buffer transmission is stopped.<br>1: UART Buffer transmission is started. |
| 0 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |

#### RTS Bit (UART Buffer Transmission Start)

When transmitting data from the UART multi-byte data, set this bit to “1”.
Only 1 can be written to this bit; 0 cannot be written.
Write to this bit when the UTOE bit in the RLN3nLUOER register is 1 (transmission enable) and the UTS bit in the RLN3nLST register is 0 (transmission is not in progress).
Once set, regardless of errors, this bit is automatically cleared to 0 upon completion of the number of data transmission specified by the MDL bit in the RLN3nLDFC register. Moreover, this bit is automatically cleared to 0 upon transition to reset mode.
Writing a value to this bit is disabled when the OMM0 bit of the RLN3nLMST register is $0_B$ (in LIN reset mode).
When writing 1 to this bit while the UTSW bit in the RLN3nLDFC register is 1: start of transmission of multi-byte UART buffers is delayed until the stop bit of reception data is completed.

### 18.3.4.9 RLN3nLMST — UART Mode Status Register (n = 0 to 3)

**Access:** This register can be read only in 8-bit units

**Address:** <RLIN3n_base> + $11_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | OMM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 18.56 RLN3nLMST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is returned.<br>Writing is ignored. |
| 0 | OMM0 | LIN Reset Status Monitor<br>0: The module is in LIN reset mode.<br>1: The module is not in LIN reset mode. |

#### OMM0 Bit (LIN Reset Status Monitor)

The OMM0 bit indicates the current operating mode.

It takes 3 peripheral clock cycles + 4 communication clock cycles, until the OMM0 bit in the RLN3nLMST register reflects the OM0 bit setting in the RLN3nLCUC register.

### 18.3.4.10 RLN3nLST — UART Status Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $12_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | URS | UTS | ERR | — | — | FTC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R | R | R/W |

**Table 18.57 RLN3nLST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 5 | URS | Data Reception Status<br>0: Reception is not in progress.<br>1: Reception is in progress. |
| 4 | UTS | Transmission Status<br>0: Transmission is not in progress.<br>1: Transmission is in progress. |
| 3 | ERR | Error Detection Flag<br>0: No error has been detected.<br>1: Error has been detected. |
| 2, 1 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 0 | FTC | Successful UART Multi-byte Transmission Flag<br>0: UART multi-byte transmission has not been completed.<br>1: UART multi-byte transmission has been completed. |

The RLN3nLST register is automatically cleared to "$00_H$" upon transition to LIN reset mode. In LIN reset mode, this register cannot be written to. In LIN reset mode, the register retains "$00_H$". To clear the specific bits in the register, write 0 to the bits to be cleared and write 1 to the other bits using the single memory storing instruction.

#### URS Bit (Data Reception Status)

At the start of the reception, this flag is set to 1.
The reception is started under the following condition.

- When the start bit is detected

At the end of reception, this flag is cleared to 0. While reception is stopped, this flag retains 0.
The reception is ended under the following conditions.

- Sampling point of the first bit of the stop bits

#### UTS Bit (Transmission Status Flag)

At the start of the transmission, this flag is set to 1. During the transmission, this flag retains 1.

The transmission is started under the following conditions.

- When transmission data is set to the RLN3nLUTDR or RLN3nLUWTDR register

- When the RTS bit in the RLN3nLTRC register is set to 1

This flag is cleared to 0 at the end of transmission.
The transmission is ended under the following conditions.

- When transmission of data set in the RLN3nLUTDR or RLN3nLUWTDR register is completed and next data is not set
- When the UART multi-byte transmission is completed (when the RTS bit in the RLN3nLTRC register is cleared to 0)
- When transmission operation enable bit UTOE in register LUOER is cleared.

#### ERR Flag (Error Detection Flag)

This flag is set to 1 upon detection of an error, detection of an expansion bit, or upon ID matching (when the value of at least one of the flags of the RLN3nLEST register is 1). At this time, an interrupt is generated. However, when this bit is 1, an interrupt is not generated upon detection of an error, detection of an expansion bit, or upon ID matching. To clear the bit to 0, write 0 to the UPER, EXBT, OER, and ER flags in the RLN3nLEST register.

#### FTC Flag (Successful Frame/Wake-up Transmission Flag)

Only 0 can be written to the FTC flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
Regardless of errors, this bit is set to 1 upon completion of the number of data transmission specified by the MDL bit in the RLN3nLDFC register, when the UART multi-byte mode is used. At this time, an interrupt is generated.

### 18.3.4.11 RLN3nLEST — UART Error Status Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $13_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | UPER | IDMT | EXBT | FER | OER | — | BER |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R | R/W |

**Table 18.58 RLN3nLEST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 6 | UPER | Parity Error Flag<br>0: Parity error has not been detected.<br>1: Parity error has been detected. |
| 5 | IDMT | ID Matching Flag<br>0: The receive data does not match with the ID value.<br>1: The receive data matches with the ID value. |
| 4 | EXBT | Expanded Bit Detection Flag<br>0: Expanded bit has not been detected.<br>1: Expanded bit has been detected. |
| 3 | FER | Framing Error Flag<br>0: Framing error has not been detected.<br>1: Framing error has been detected. |
| 2 | OER | Overrun Error Flag<br>0: Overrun error has not been detected.<br>1: Overrun error has been detected. |
| 1 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 0 | BER | Bit Error Flag<br>0: Bit error has not been detected.<br>1: Bit error has been detected. |

The RLN3nLEST register is automatically cleared to $00_H$ when the module transitions to LIN reset mode. In LIN reset mode, this register cannot be written to, and the value of $00_H$ is held. To clear certain bits in this register, write 0 to those bits, and write 1 to the bits not to be cleared by using the single memory storing instruction.

#### UPER Flag (Parity Error Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
This flag is set to 1 upon parity error detection. To clear the bit, write 0 to the bit.

#### IDMT Flag (ID Matching Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value that has been retained before 1 is written.

The IDMT flag becomes 1 when all the following conditions are met:

- The UEBE bit in the RLN3nLUOR1 register is 1 (expansion bit enabled)
- The UECD bit in the RLN3nLUOR1 register is 0 (expansion bit comparison enabled)
- The UEBDCE bit in the RLN3nLUOR1 register is 1 (expansion bit/data comparison enabled)
  - The received expansion bit and the value of the UEBDL bit of the RLN3nLUOR1 register are matched.
  - The 8-bit receive data excluding the expansion bit and the value of the RLN3nLIDB register are matched.

To clear the bit, write 0 to the bit.

#### EXBT Flag (Expanded Bit Detection Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
When the UEBE bit in the RLN3nLUOR1 register is 1 (expansion bit enable), if the received expansion bit matches with the UEBDL bit in the RLN3nLUOR1 register, this flag is set to 1.
To clear the bit, write 0 to the bit.

#### FER Flag (Framing Error Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The FER flag is set to 1 upon framing error detection while the FERE bit of the RLN3nLEDE register is 1 (framing error detection enabled). To clear the bit, write 0 to the bit.
Framing error is detected always on the first stop bit, regardless of the stop bit amount setting.

#### OER Flag (Overrun Error Flag)

Only 0 can be written to this flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
This flag is set to 1 upon overrun error detection. This is, when receive data has been stored into the UART receive data register and the next receive operation is completed before that receive data has been read.

#### BER Flag (Bit Error Flag)

Only 0 can be written to the BER flag; when 1 is written, the bit retains the value that has been retained before 1 is written.
The BER flag is set to 1 when the transmitted data and the data monitored at the sampling point position at the receive input do not match while the BERE bit of the RLN3nLEDE register is 1 (bit error detection enabled).
To clear the bit, write 0 to the bit.

### 18.3.4.12 RLN3nLDFC — UART Data Field Configuration Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units.

**Address:** <RLIN3n_base> + $14_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | UTSW | — | MDL[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R | R/W | R/W | R/W | R/W |

**Table 18.59 RLN3nLDFC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 5 | UTSW | Multi-byte Transmission Start Wait<br>0: When UART multi-byte transmission is requested, transmission is started immediately.<br>1: When UART multi-byte transmission is requested, transmission is not started until reception of the stop bit is completed. |
| 4 | Reserved | When read, the value after reset is returned.<br>When writing to this bit, write the value after reset. |
| 3 to 0 | MDL[3:0] | UART multi-byte Data Length Select<br>b3 b0<br>0 0 0 0: 9 bytes<br>0 0 0 1: 1 bytes<br>0 0 1 0: 2 bytes<br>0 0 1 1: 3 bytes<br>0 1 0 0: 4 bytes<br>0 1 0 1: 5 bytes<br>0 1 1 0: 6 bytes<br>0 1 1 1: 7 bytes<br>1 0 0 0: 8 bytes<br>1 0 0 1: 9 bytes<br>1 0 1 0: 9 bytes<br>:<br>1 1 1 1: 9 bytes |

#### UTSW Bit (Transmission Start Wait)

This bit controls the transmission start timing of UART multi-byte data.
With 0 set, transmission is started as soon as the start of UART multi-byte data transmit is requested.
With 1 set, transmission is started after the completion of the stop bit reception.
Note that the wait time is only 1 bit even if the stop bit length is set to 2 bits with the USBLS bit in the RLN3nLBFC register.
This bit is enabled when the RTS bit in the RLN3nLTRC register is set to 1. In addition, writing a value to this bit is disabled when the RTS bit is 1 (UART multi-byte data transmission started).
Set this bit to 1 only to switch from reception to transmission in half-duplex communication.

#### MDL[3:0] Bits (UART Buffer Data Length Select)

This bit specifies the data length of the UART multi-byte data.
Writing a value to these bits is disabled when the RTS bit in the RLN3nLTRC register is 1 (UART multi-byte data transmission started).

### 18.3.4.13 RLN3nLIDB — UART ID Buffer Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $15_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ID[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.60 RLN3nLIDB register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | ID[7:0] | ID value that is referred for the expansion bit data comparison is set |

#### ID Bit (ID Bit)

When the UEBE bit in the RLN3nLUOR1 register is set to 1 (expansion bit enabled), the UECD bit is set to 0 (expansion bit comparison enabled), and the UEBDCE bit is set to 1 (expansion bit/data comparison enabled), write the reference ID value for comparison into this register. Write to the RLN3nLIDB register when the URS bit in the RLN3nLST register is 0 (receive operation is not in progress).

### 18.3.4.14 RLN3nLUDB0 — UART Data 0 Buffer Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $17_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | UDB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.61 RLN3nLUDB0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | UDB[7:0] | Value of UART data |

If the data length selection corresponds to 9 data bytes (RLN3nLDFC.MDL bit is "$0_H$" or "$9_H$") for multi-byte UART transmission, then the first data value for UART communication is present in this buffer.

Write to the RLN3nLUDB0 register when the RTS bit of the RLN3nLTRC register is 0 (UART multi-byte data transmission stopped).

For details about the UART multi-byte data, see **Section 18.5.4.1 (2), UART Buffer Transmission**.

### 18.3.4.15 RLN3nLDBRb — UART Data Buffer b Register (n = 0 to 3, b = 1 to 8)

**Access:** This register can be read/written in 8-bit units.

**Address:** RLN3nLDBR1: <RLIN3n_base> +$18_H$
RLN3nLDBR2: <RLIN3n_base> + $19_H$
RLN3nLDBR3: <RLIN3n_base> + $1A_H$
RLN3nLDBR4: <RLIN3n_base> + $1B_H$
RLN3nLDBR5: <RLIN3n_base> + $1C_H$
RLN3nLDBR6: <RLIN3n_base> + $1D_H$
RLN3nLDBR7: <RLIN3n_base> + $1E_H$
RLN3nLDBR8: <RLIN3n_base> + $1F_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LDB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.62 RLN3nLDBRb register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | LDB[7:0] | Sets the data to be transmitted. |

This register specifies the data transmitted in UART multi-byte mode.

Write to these registers when the RTS bit of the RLN3nLTRC register is 0 (UART multi-byte data transmission stopped).

For details about the UART multi-byte data, see **Section 18.5.4.1(2), UART Buffer Transmission**.

### 18.3.4.16 RLN3nLUOER — UART Operation Enable Register (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | UROE | UTOE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 18.63 RLN3nLUOER register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 1 | UROE | Reception Enable<br>0: Disables reception.<br>1: Enables reception. |
| 0 | UTOE | Transmission Enable<br>0: Disables transmission.<br>1: Enables transmission. |

The RLN3nLUOER register is automatically cleared to $00_H$ upon transition to LIN reset mode.
In LIN reset mode, this register cannot be written to.
In LIN reset mode, the register retains $00_H$.

#### UROE Bit (Reception Enable)

The UROE bit enables or disables reception.
With 0 set, reception is disabled.
With 1 set, reception is enabled.
Do not clear this bit during reception and not set this bit during multi-byte transmission. If the communication is suspended during reception, set the OM0 bit in the RLN3nLCUC register to 0 (LIN reset mode) to shift to the LIN reset mode. However, the transmit operation is also suspended at this time.

#### UTOE Bit (Transmission Enable)

The UTOE bit enables or disables transmission.
With 0 set, transmission is disabled.
With 1 set, transmission is enabled.
Do not clear this bit during transmission. If the communication is suspended during transmission, set the OM0 bit in the RLN3nLCUC register to 0 (LIN reset mode) to shift to the LIN reset mode. However, the receive operation is also suspended at this time.

### 18.3.4.17 RLN3nLUOR1 — UART Option Register 1 (n = 0 to 3)

**Access:** This register can be read/written in 8-bit units

**Address:** <RLIN3n_base> + $21_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | UECD | UTIGTS | UEBDCE | UEBDL | UEBE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 18.64 RLN3nLUOR1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 4 | UECD | Expansion Bit Comparison Disable<br>0: Enables comparison between the received expansion bit and the UEBDL bit value.<br>1: Disables comparison between the received expansion bit and the UEBDL bit value. |
| 3 | UTIGTS | Transmission Interrupt Generation Timing Select<br>0: Transmission interrupt is generated at the start of transmission.<br>1: Transmission interrupt is generated at the completion of transmission. |
| 2 | UEBDCE | Expansion Bit Data Comparison Enable<br>0: Disables data comparison after an expansion bit is detected.<br>1: Enables data comparison after an expansion bit is detected. |
| 1 | UEBDL | Expansion Bit Detection Level Select<br>0: Selects expansion bit value 0 as the expansion bit detection level.<br>1: Selects expansion bit value 1 as the expansion bit detection level. |
| 0 | UEBE | Expansion Bit Enable<br>0: Disables expansion bit operation.<br>1: Enables expansion bit operation. |

#### UECD Bit (Expansion Bit Comparison Disable)

The UECD bit enables or disables comparison between the received expansion bit and the UEBDL bit value when the UEBE bit is 1 (expansion bit operation is enabled).
With 0 set, comparison between the received expansion bit and the UEBDL bit value is enabled when the expansion bit is received.
With 1 set, comparison between the received expansion bit and the UEBDL bit value is disabled when the expansion bit is received.
Set this bit when the OMM0 bit of the RLN3nLMST register is $0_B$ (in LIN reset mode).
Do not set this bit to 1 when the UART multi-byte mode is used.
Do not set this bit to 1 when the UEBDCE bit is 1 (expansion bit/data comparison enable).

#### UTIGTS Bit (Transmission Interrupt Generation Timing Select)

The UTIGTS bit sets the generation timing of the transmission interrupt.
With 0 set, the transmission interrupt is generated at the start of transmission.
With 1 set, the transmission interrupt is generated at the completion of transmission.
When transmission from the UART multi-byte mode is performed with 0 set, the transmission interrupt is generated only at the start of the transmission of the last data of the data length set with the MDL bits

in the RLN3nLDFC register.
When transmission from the UART multi-byte mode is performed with 1 set, the transmission interrupt is generated only at the completion of the transmission of the last data of the data length set with the MDL bits in the RLN3nLDFC register.

#### UEBDCE Bit (Expansion Bit Data Comparison Enable)

After an expansion bit is detected, this bit enables or disables the comparison between the 8-bit receive data excluding the expansion bit and the value of the RLN3nLIDB register.
With 0 set, when the level selected by the UEBDL bit is detected as an expansion bit, the comparison between the receive value in the RLN3nLURDR register and the value of the RLN3nLIDB register is disabled.
With 1 set, when the level selected by the UEBDL bit is detected as an expansion bit, the comparison between the receive value in the RLN3nLURDR register and the value of the RLN3nLIDB register is enabled.
Set this bit when the OMM0 bit of the RLN3nLMST register is $0_B$ (in LIN reset mode).
Do not set this bit to 1 when the UEBE bit is 0 (expansion bit operation disabled).
Do not set this bit to 1 when the UECD bit is 1 (expansion bit comparison disabled).
Do not set this bit to 1 when the UART multi-byte mode is used.

#### UEBDL Bit (Expansion Bit Detection Level Select)

The UEBDL bit sets the level to be detected as the expansion bit when the UEBE bit is 1 (expansion bit operation is enabled) and the UECD bit is 0 (comparison of the expansion bit is enabled).
With 0 set, expansion bit value 0 is the level to be detected as the expansion bit.
With 1 set, expansion bit value 1 is the level to be detected as the expansion bit.
Set this bit when the OMM0 bit of the RLN3nLMST register is $0_B$ (in LIN reset mode).
Do not set this bit to 1 when the UART multi-byte mode is used.

#### UEBE Bit (Expansion Bit Enable Bit)

The UEBE bit enables or disables expansion bit operation.
With 0 set, expansion bit operation is disabled.
With 1 set, expansion bit operation is enabled.
Set this bit when the OMM0 bit of the RLN3nLMST register is $0_B$ (in LIN reset mode).
Do not set this bit to 1 when the UART multi-byte mode is used.

### 18.3.4.18 RLN3nLUTDR — UART Transmission Data Register (n = 0 to 3)

**Access:** RLN3nLUTDR register can be read/written in 16-bit units.
RLN3nLUTDRL register can be read/written in 8-bit units.
RLN3nLUTDRH register can be read/written in 8-bit units.

**Address:** RLN3nLUTDR: <RLIN3n_base> + $24_H$ (n = 0 to 3)
RLN3nLUTDRL: <RLIN3n_base> + $24_H$ (n = 0 to 3)
RLN3nLUTDRH: <RLIN3n_base> + $25_H$ (n = 0 to 3)

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | UTD[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.65 RLN3nLUTDR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 9 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 8 to 0 | UTD[8:0] | Sets the data to be transmitted from the transmission buffer.<br>Setting Range: $000_H$ to $1FF_H$ |

The RLN3nLUTDR register sets the data to be transmitted from the transmit data register.

Writing data to this register with the UTOE bit in the RLN3nLUOER register set to 1 starts transmission.

This register can be accessed in 8 bits.

In 9-bit communication mode, do not attempt 8-bit access.

Do not write data to this register when data transmission from the UART multi-byte data is in progress.

Also, do not write data to this register when a transmission request is being generated due to write access to the RLN3nLUWTDR register.

When transmitting data continuously, do not set another piece of transmission data in this register before the generation of transmission interrupt.

The table below shows the bit arrangement according to the set communication format.

**Table 18.66 Bit Arrangement of the RLN3nLUTDR Register According to Each Communication Format**

| | RLN3nLUTDR | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 7-bit; LSB first | — | — | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 7-bit; MSB first | — | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 |
| 8-bit; LSB first | — | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 8-bit; MSB first | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 |
| 9-bit; LSB first | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 9-bit; MSB first | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 | Bit8 |

### 18.3.4.19 RLN3nLURDR — UART Reception Data Register (n = 0 to 3)

**Access:** RLN3nLURDR register can be read only in 16-bit units.
RLN3nLURDRL register can be read only in 8-bit units.
RLN3nLURDRH register can be read only in 8-bit units.

**Address:** RLN3nLURDR: <RLIN3n_base> + $26_H$ (n = 0 to 3)
RLN3nLURDRL: <RLIN3n_base> + $26_H$ (n = 0 to 3)
RLN3nLURDRH: <RLIN3n_base> + $27_H$ (n = 0 to 3)

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | URD [8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 18.67 RLN3nLURDR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 9 | Reserved | When read, the value after reset is returned.<br>Writing is ignored. |
| 8 to 0 | URD [8:0] | Value of received data<br>Setting Range: $000_H$ to $1FF_H$ |

The RLN3nLURDR allows the reception data to be read from the receive data register.

When the UROE bit in the RLN3nLUOER register is 1, the reception data is stored in this register and can be read out.

This register is updated at the stop bit of the reception data.

This register is also updated when an error is caused by the parity or stop bit.

However, the value of this register is not updated upon occurrence of an overrun error when the OERE bit of the RLN3nLEDE register is 1 (overrun detection enabled). The value of this register is updated upon occurrence of an overrun error when the OERE bit is 0 (overrun detection disabled).

Read this register upon occurrence of a receive error (overrun error, framing error, parity error) when the OERE bit of the RLN3nLEDE register is 1 (overrun error detection enabled). If the next data is received without reading this register, an overrun error occurs.

This register can be accessed in 8 bits.

However, during expansion bit use (UEBE bit of the RLN3nLUOR1 register is 1 (expansion bit operation enabled), do not attempt 8-bit access.

The table below shows the bit arrangement according to the set communication format.

**Table 18.68 Bit Arrangement of the RLN3nLURDR Register According to Each Communication Format**

| | RLN3nLURDR | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **b8** | **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| 7-bit; LSB first | — | — | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 7-bit; MSB first | — | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 |
| 8-bit; LSB first | — | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 8-bit; MSB first | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 |
| 9-bit; LSB first | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 9-bit; MSB first | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 | Bit8 |

### 18.3.4.20 RLN3nLUWTDR — UART Wait Transmission Data Register (n = 0 to 3)

**Access:** RLN3nLUWTDR register can be read/written in 16-bit units.
RLN3nLUWTDRL register can be read/written in 8-bit units.
RLN3nLUWTDRH register can be read/written in 8-bit units.

**Address:** RLN3nLUWTDR: <RLIN3n_base> + $28_H$ (n = 0 to 3)
RLN3nLUWTDRL: <RLIN3n_base> + $28_H$ (n = 0 to 3)
RLN3nLUWTDRH: <RLIN3n_base> + $29_H$ (n = 0 to 3)

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | UWTD[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 18.69 RLN3nLUWTDR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 9 | Reserved | When read, the value after reset is returned.<br>When writing to these bits, write the value after reset. |
| 8 to 0 | UWTD[8:0] | Sets the data to be transmitted from the wait transmit data register after waiting for the stop bit reception to be completed.<br>Setting Range: $000_H$ to $1FF_H$ |

The RLN3nLUWTDR register sets the data to be transmitted from the UART wait transmit data register.

Writing data to this register with the UTOE bit in the RLN3nLUOER register set to 1 starts transmission.

Use this register only to switch from reception to transmission in half-duplex communication.

The user should use this register only if data reception is expected or in progress..

Note that the wait time is only 1 bit even if the stop bit length is set to 2 bits with the USBLS bit in the RLN3nLBFC register.

When this register is read, the RLN3nLUTDR register value is actually read.

In 9-bit communication mode, do not attempt 8-bit access.

Do not write data to this register when data transmission from the UART multi-byte data is in progress.

The table below shows the bit arrangement according to the set communication format.

**Table 18.70 Bit Arrangement of the RLN3nLUWTDR Register According to Each Communication Format**

| | RLN3nLUWTDR | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 7-bit; LSB first | — | — | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 7-bit; MSB first | — | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 |
| 8-bit; LSB first | — | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 8-bit; MSB first | — | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 |
| 9-bit; LSB first | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 9-bit; MSB first | Bit0 | Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 | Bit8 |

## 18.4 Interrupt Sources

The LIN/UART interface generates four types of interrupt requests.

- RLIN3n successful transmission interrupt
- RLIN3n successful reception interrupt
- RLIN3n status interrupt

It is required to set the LIOS bit in the RLN3nLMD register to 1 in order to output the RLIN3n transmission interrupt, RLIN3n successful reception interrupt, or RLIN3n status interrupt depending on the interrupt request.

**Table 18.71** lists the sources for each interrupt.

**Table 18.71 Interrupt Sources**

| | | LIOS bit in RLN3nLMD register is 1 *1 | | | |
|---|---|---|---|---|---|
| | | **RLIN3n Interrupt** | **RLIN3n Transmission Interrupt** | **RLIN3n Successful Reception Interrupt** | **RLIN3n Status Interrupt** |
| LIN mode | LIN master mode | • Successful frame transmission<br>• Successful frame reception<br>• Successful wake-up transmission<br>• Successful wake-up reception<br>• Successful header transmission<br>• Bit error<br>• Physical bus error<br>• Frame/response timeout error<br>• Framing error<br>• Checksum error<br>• Response preparation error | • Successful frame transmission<br>• Successful wake-up transmission<br>• Successful header transmission | • Successful frame reception<br>• Successful wake-up reception | • Bit error<br>• Physical bus error<br>• Frame/response timeout error<br>• Framing error<br>• Checksum error<br>• Response preparation error |
| | LIN slave mode | • Successful response transmission<br>• Successful response reception<br>• Successful wake-up transmission<br>• Successful wake-up reception<br>• Successful header reception<br>• Bit error<br>• Frame/response timeout error<br>• Framing error<br>• Sync field error<br>• Checksum error<br>• ID parity error<br>• Response preparation error | • Successful response transmission<br>• Successful wake-up transmission | • Successful response reception<br>• Successful wake-up reception<br>• Successful header reception | • Bit error<br>• Frame/response timeout error<br>• Framing error<br>• Sync field error<br>• Checksum error<br>• ID parity error<br>• Response preparation error |
| UART mode | | — | • Transmission start/successful transmission | • Successful reception<br>• Expansion bit mismatch | • Bit error<br>• Overrun error<br>• Framing error<br>• Expansion bit detection<br>• ID match<br>• Parity error |

Note 1. The LIOS bit setting is valid in LIN Mode. In UART mode, setting the LIOS bit is not required.

Each interrupt request is output when the corresponding bit in the RLN3nLIE register is 1 (interrupt is enabled) and the corresponding flag in the RLN3nLST register is 1.

## 18.5 Operation

### 18.5.1 Modes

The LIN/UART interface provides the following four modes, depending upon the specific function to be performed:

- LIN reset mode
- LIN mode
  (LIN master mode/LIN slave mode [auto baud rate]/LIN slave mode [fixed baud rate])
- UART mode
- LIN self-test mode

The supply of clocks to the LIN/UART interface is stopped in LIN reset mode, which reduces power consumption.

**Figure 18.2** shows mode transitions. **Table 18.72** describes mode transition conditions. **Table 18.73** lists operations available in each mode.

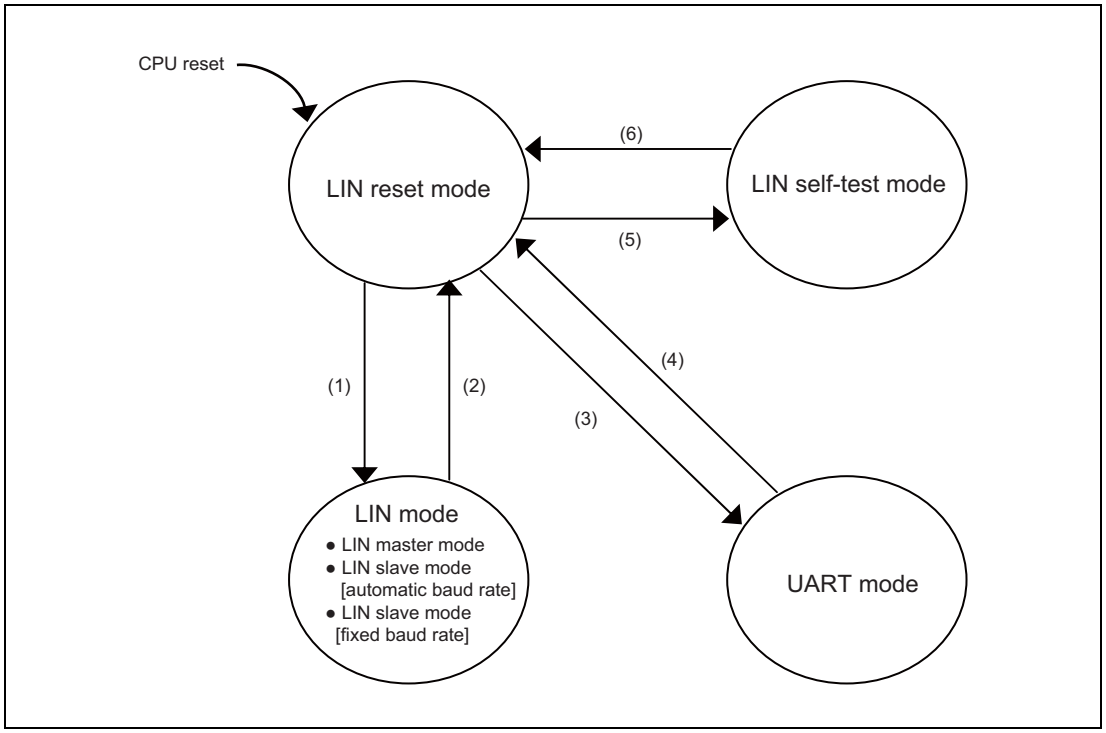



**Figure 18.2 Mode Transitions**

**Table 18.72 Transition Condition of Each Mode**

| | Mode Transition | | | Transition Condition |
|---|---|---|---|---|
| 1 | LIN reset mode | → | LIN mode<br>(LIN master mode) | LMD bit in RLN3nLMD register = $00_B$<br>and<br>OM1 and OM0 bits in RLN3nLCUC register = $01_B$ or $11_B$ |
| | LIN reset mode | → | LIN mode<br>(LIN slave mode [auto baud rate]) | LMD bit in RLN3nLMD register = $10_B$<br>and<br>OM1 and OM0 bits in RLN3nLCUC register = $01_B$ or $11_B$ |
| | LIN reset mode | → | LIN mode<br>(LIN slave mode [fixed baud rate]) | LMD bit in RLN3nLMD register = $11_B$<br>and<br>OM1 and OM0 bits of RLN3nLCUC register = $01_B$ or $11_B$ |
| 2 | LIN mode | → | LIN reset mode | OM0 bit in RLN3nLCUC register = $0_B$ |
| 3 | LIN reset mode | → | UART mode | LMD bit in RLN3nLMD register = $01_B$<br>and<br>OM0 bit in RLN3nLCUC register = $1_B$ |
| 4 | UART mode | → | LIN reset mode | OM0 bit in RLN3nLCUC register = $0_B$ |
| 5 | LIN reset mode | → | LIN self-test mode | See **Section 18.5.5, LIN Self-Test Mode**. |
| 6 | LIN self-test mode | → | LIN reset mode | See **Section 18.5.5, LIN Self-Test Mode**. |

**Table 18.73 Operations Available in Each Mode**

| LIN mode | | | |
|---|---|---|---|
| **LIN master mode** | **LIN slave mode [auto baud rate]<br>LIN slave mode [fixed baud rate]** | **UART mode** | **LIN self-test mode** |
| Header transmission<br>Response transmission<br>Response reception<br>Wake-up transmission<br>Wake-up reception<br>Error detection | Header reception<br>Response transmission<br>Response reception<br>Wake-up transmission<br>Wake-up reception<br>Error detection | UART transmission<br>UART reception<br>Error detection | Self test |

Whether a transition has been caused to LIN reset mode, the LIN mode, or the UART mode can be verified by reading the LMD bits in the RLN3nLMD register or the OMM0 bit in the RLN3nLMST register.

For a description of the LIN self-test mode, see **Section 18.5.5, LIN Self-Test Mode**.

### 18.5.2 LIN Reset Mode

Setting the OM0 bit in the RLN3nLCUC register to 0 (LIN reset mode) causes a transition to LIN reset mode. The change to LIN reset mode can be verified by determining that the OMM0 bit in the RLN3nLMST register has been set to 0 (LIN reset mode). In this mode, the LIN communication and the UART communication functions are halted.

From LIN reset mode, transitions to LIN mode, UART mode, and LIN self-test mode can be made.

When the mode changes to LIN reset mode, the following registers are initialized to their reset values, and as long as LIN reset mode is in effect, they retain their initial values.

- RLN3nLTRC register
- RLN3nLST register
- RLN3nLEST register
- RLN3nLUOER register

The following registers retain their previous values even when a transition to LIN reset mode is made:

- RLN3nLWBR register
- RLN3nLBRP0 register
- RLN3nLBRP1 register
- RLN3nLMD register
- RLN3nLBFC register
- RLN3nLSC register
- RLN3nLWUP register
- RLN3nLIE register
- RLN3nLEDE register
- RLN3nLDFC register
- RLN3nLIDB register
- RLN3nLCBR register
- RLN3nLUDB0 register
- RLN3nLDBRb register (b = 1 to 8)
- RLN3nLUOR1 register
- RLN3nLUTDR register
- RLN3nLURDR register
- RLN3nLUWTDR register

### 18.5.3 LIN Mode

The LIN mode provides the following three operation modes:

- LIN operation mode
- LIN wake-up mode
- LIN self-test mode

LIN mode can operate in the following submodes: LIN master mode, LIN slave mode [auto baud rate], and LIN slave mode [fixed baud rate].

In LIN master mode, the following operations can be performed: header transmission, response transmission, response reception, wake-up transmission, wake-up reception, and error detection. In LIN reset mode, setting the LMD bits in the RLN3nLMD register to $00_B$ (LIN master mode) and the OM1 and OM0 bits in the RLN3nLCUC register to either $01_B$ or $11_B$ sets LIN master mode, turning the OMM1 and OMM0 bits in the RLN3nLMST register to either $01_B$ to $11_B$.

In LIN slave mode [auto baud rate] and LIN slave mode [fixed baud rate], header reception, response transmission, response reception, wake-up transmission, wake-up reception, and error detection can be performed.

The LIN slave mode [auto baud rate] allows automatic detection of the break field and the sync field, and sets a baud rate based on the results of measurement of a sync field. The baud rate can be set to the range from 1kbps to 20kbps.

Set the LPRS[2:0] bits in the RLN3nLWBR register so that the prescaler clock (with the frequency of the LIN communication clock source divided by the prescaler) becomes as follows according to the target baud rate.

| [Target baud rate] | [Prescaler clock] |
|---|---|
| 1 to 20 kbps | : 4 MHz*1 |
| 1 to 2.4 kbps (excluding 2.4 kbps) | : 4 MHz |
| 2.4 to 20 kbps | : 8 to 12 MHz |

Note 1. Use the clock with NSPB[3:0] bits in the RLN3nLWBR register set to "$0011_B$" (four samplings).

In LIN slave mode [fixed baud rate] allows automatic detection of the break field, the sync field, and the ID filed at a baud rate that is set in advance by the baud rate generator.

In LIN reset mode, setting the LMD bits in the RLN3nLMD register to $10_B$ (LIN slave mode [auto baud rate] and setting the OM1 and OM0 bit in the RLN3nLCUC register to $01_B$ or $11_B$ sets LIN slave mode [auto baud rate]; and setting the LMD bits in the RLN3nLMD register to $11_B$ (LIN slave mode [fixed baud rate]), and setting the OM1 and OM0 bits in the RLN3nLCUC register to $01_B$ or $11_B$ sets LIN slave mode [fixed baud rate], turning the OMM1 and OMM0 bits in the RLN3nLMST register to $01_B$ or $11_B$.

When changing a submode to another submode within LIN mode, a transition to LIN reset mode should first be made and change the LMD bits in the RLN3nLMD register.

The LIN mode provides the following two operation modes:

- LIN operation mode
- LIN wake-up mode

**Figure 18.3** shows the transition of operation modes. **Table 18.74** describes the transition conditions of operation modes.

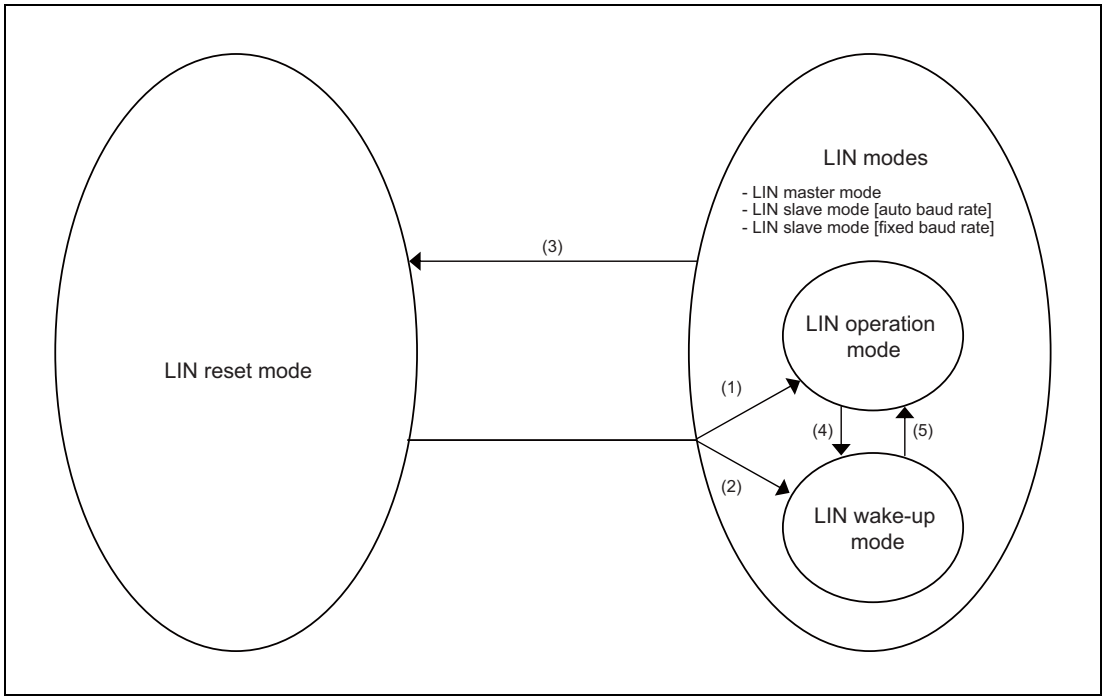



**Figure 18.3 Transition of Operation Modes**

**Table 18.74 Transition condition for Operation Mode**

| | Operation mode transition | | | Transition condition |
|---|---|---|---|---|
| (1) | LIN reset mode | → | LIN mode<br>- LIN operation mode | LMD bit in RLN3nLMD register = $00_B$ or $10_B$ or $11_B$<br>and<br>OM1 and OM0 bits in RLN3nLCUC register = $11_B$ |
| (2) | LIN reset mode | → | LIN mode<br>- LIN wake-up mode | LMD bit in RLN3nLMD register = $00_B$ or $10_B$ or $11_B$<br>and<br>OM1 and OM0 bits in RLN3nLCUC register = $01_B$ |
| (3) | LIN mode<br>- LIN operation mode<br>- LIN wake-up mode | → | LIN reset mode | OM0 bit in LCUC register = $0_B$ |
| (4) *1 | LIN mode<br>- LIN operation mode | → | LIN mode<br>- LIN wake-up mode | OM1 and OM0 bits in RLN3nLCUC register = $01_B$ |
| (5) *1 | LIN mode<br>- LIN wake-up mode | → | LIN mode<br>- LIN operation mode | OM1 and OM0 bits in RLN3nLCUC register = $11_B$ |

Note 1. Transition between LIN operation mode and LIN wake-up mode cannot be made when communication is going on (when the FTS bit in the RLN3nLTRC register is 1).

#### (1) LIN Operation Mode

In LIN operation mode, frame processing (header transmission, header reception, response transmission, response reception, and error detection) can be performed.

During a transition from LIN reset mode to LIN mode, setting the OM1 and OM0 bits in the RLN3nLCUC register to $11_B$ changes the mode to LIN operation mode, changing the OMM1 and OMM0 bits in the RLN3nLMST register to $11_B$. Communication settings should be performed after the OMM1 and OMM0 bits have become $11_B$.

#### (2) LIN Wake-up Mode

In LIN wake-up mode, wake-up signal processing (wake-up transmission, wake-up reception, and error detection) can be performed.

During a transition from LIN reset mode to LIN mode, setting the OM1 and OM0 bits in the RLN3nLCUC register to $01_B$ changes the mode to LIN wake-up mode, changing the OMM1 and OMM0 bits in the RLN3nLMST register to $01_B$. Communication settings should be performed after the OMM1 and OMM0 bits have become $01_B$.

### 18.5.3.1 LIN Master Mode

#### (1) Header Transmission

**Figure 18.4** shows the operation of the LIN/UART interface (LIN master mode) in header transmission. **Table 18.75** provides processing in header transmission.

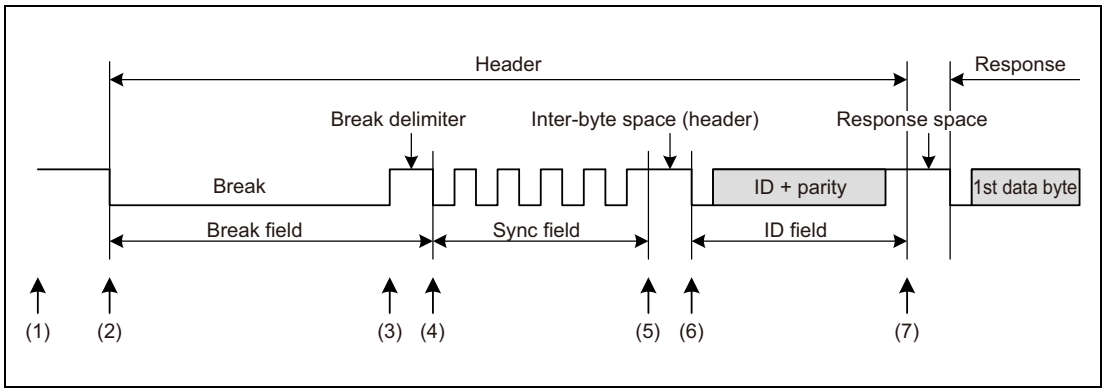



**Figure 18.4 Operation in Header Transmission**

**Table 18.75 Processing in Header Transmission (1/2)**

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (1) | • Sets a baud rate<br>• Sets noise filter ON/OFF<br>• Enables interrupt<br>• Enables error detection<br>• Sets frame configuration parameters<br>• Changes the LIN/UART interface to the LIN master mode: LIN operation mode<br>• Sets information on the frame to be transmitted (ID, parity, data length, response direction, Checksum method, and transmission data) | Waits for the setting of the FTS bit in the RLN3nLTRC register by software (idle) |
| (2) | Sets the FTS bit in the RLN3nLTRC register to 1 (frame transmission or wake-up transmission/reception started) | Transmits a break. |

Table 18.75 Processing in Header Transmission (2/2)

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (3) | Waits for an interrupt request | Transmits a break delimiter. |
| (4) | | Transmits a sync field ($55_H$). |
| (5) | | Transmits an inter-byte space (header). |
| (6) | | Transmits an ID field. |
| (7) | | Sets a successful header transmission flag. |

**NOTE**

For information about error detection, see **Section 18.5.3.7, Error Status**.

### (2) Response Transmission

**Figure 18.5** shows the operation of the LIN/UART interface (LIN master mode) in response transmission. **Table 18.76** provides processing in response transmission.

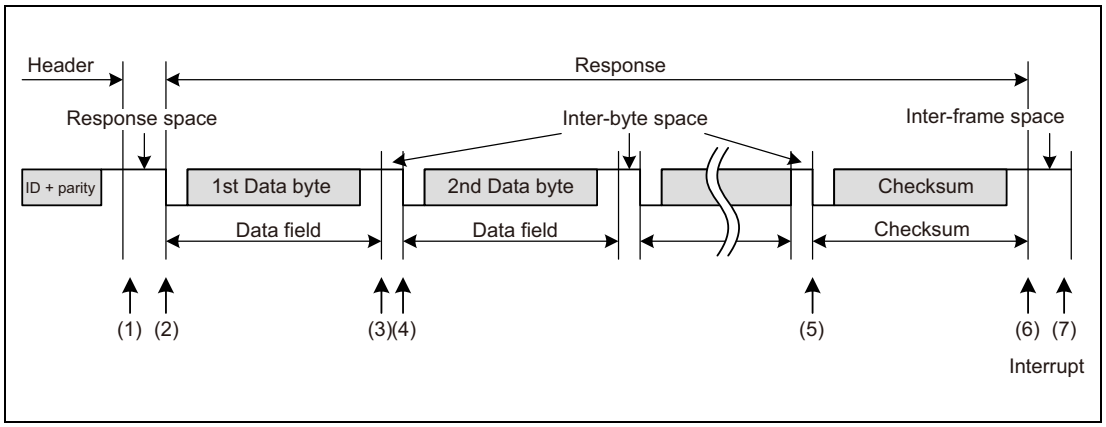



Figure 18.5 Operation in Response Transmission

Table 18.76 Processing in Response Transmission (1/2)

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (1) | (When in frame separate mode)<br>• Sets the RTS bit in the RLN3nLTRC register to 1 (response transmission/ reception started)<br>(When in frame (combined mode))<br>• Waits for an interrupt request | (When in frame separate mode)<br>• Waits for the setting of the RTS bit in the RLN3nLTRC register to 1 by software.<br>• When the bit is set to 1, sends a response space.<br>(When in frame (combined mode))<br>• Sends a response space. |

Table 18.76 Processing in Response Transmission (2/2)

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (2) | Waits for an interrupt request | Transmits 1st data byte. |
| (3) | | Transmits an inter-byte space. |
| (4) | | • Transmits 2nd data byte.<br>• Transmits an inter-byte space<br>• Transmits 3rd data byte.<br>• Transmits an inter-byte space<br>(Repeats the transmission of inter-byte spaces as many times as the data length specified in bits RFDL[3:0] in the RFC register, and stops the transmission when the BER flag in the RLN3nLEST register is 1 (bit error detected). If an error occurs, does not perform the Checksum transmission in item (5)).<br>:<br>: |
| (5) | | Transmits the checksum. |
| (6) | | • Sets a successful frame/wake-up transmission flag.<br>• Sets the FTS bit in the RLN3nLTRC register to 0 (frame transmission or wake-up transmission/reception stopped).<br>• Sets the RTS bit in the RLN3nLTRC register to 0 (when in frame separate mode, response transmission/reception stopped). |
| (7) | • Processing after communication<br>Checks the RLN3nLST register, and clears flags. | Idle |

**NOTE**

For information about error detection, see **Section 18.5.3.7, Error Status**.

### (3) Response Reception

**Figure 18.6** shows the operation of the LIN/UART interface (LIN master mode) on response reception. **Table 18.77** provides processing in response reception.

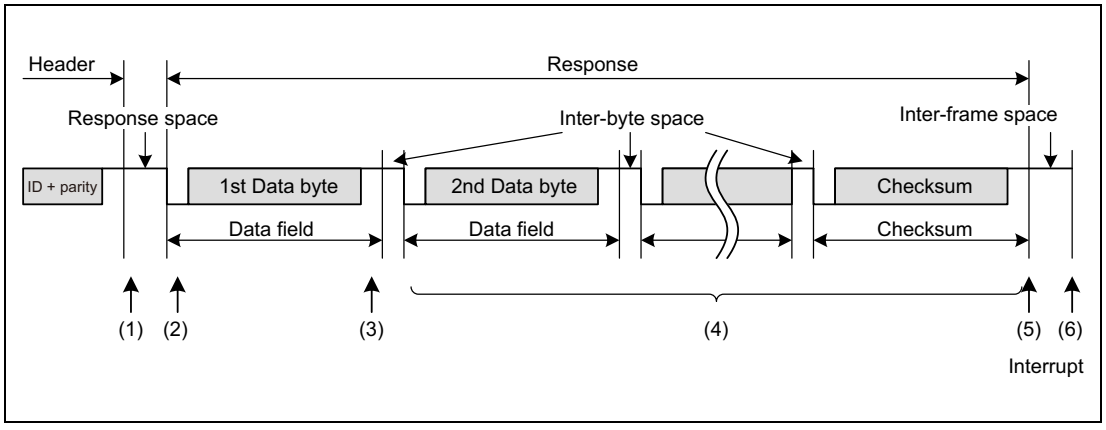



Figure 18.6 Operation in Response Reception

**Table 18.77 Processing in Response Reception**

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (1) | • When in frame separate mode<br>Sets the RTS bit in the RLN3nLTRC register to 1 (response transmission/ reception started)<br>• When in frame combined mode<br>Waits for an interrupt request (no processing) | • When in frame separate mode<br>Waits for the setting of the RTS bit in the RLN3nLTRC register to 1 by software.<br>When the bit is set to 1, waits for detection of a start bit.<br>• When in frame combined mode<br>Waits for detection of a start bit. |
| (2) | | Receives 1st data byte when the start bit is detected. |
| (3) | | Sets the successful 1st data byte reception flag. |
| (4) | | • Receives 2nd data byte when the start bit is detected.<br>• Receives 3rd data byte when the start bit is detected.<br>Repeats the reception of data bytes as many times as the data length specified in bits RFDL[3:0] in the RLN3nLDFC register, and stops the reception when any bit in the RLN3nLEST register is 1 (bit error detected). If an error occurs, the checksum determination in item (5) is not performed).<br>:<br>:<br>• Receives the checksum when the start bit is detected. |
| (5) | | • Determines the checksum.<br>• Sets the successful frame/wake-up reception flag.<br>• Sets the RTS and FTS bits in the RLN3nLTRC register to 0 (response transmission/reception stopped). |
| (6) | • Processing after communication<br>Reads the received data.<br>Checks the RLN3nLST register, and clears flags. | Idle |

**NOTE**

For information about error detection, see **Section 18.5.3.7, Error Status**.

### 18.5.3.2 LIN Slave Mode

#### (1) Header Reception

**Figure 18.7** shows the operation of the LIN/UART interface (LIN slave mode) in header reception.
**Table 18.78** provides processing in header reception.

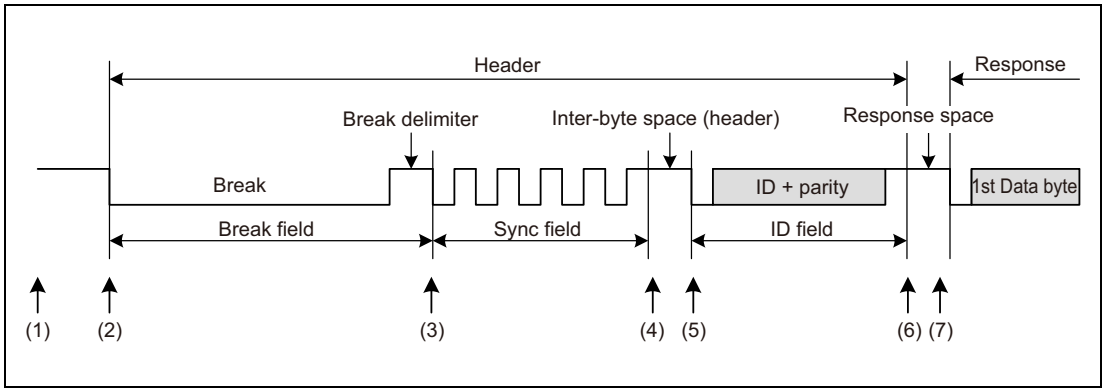



**Figure 18.7 Operation in Header Reception**

**Table 18.78 Processing in Header Reception**

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (1) | • Sets a baud rate<br>• Sets noise filter ON/OFF<br>• Enables interrupt<br>• Enables error detection<br>• Sets frame configuration parameters<br>• Changes the LIN/UART interface to the LIN slave mode: LIN operation mode<br>• Sets the FTS bit in the RLN3nLTRC register to 1 (header reception or wake-up transmission/ reception started) | Waits for the setting of the FTS bit in the RLN3nLTRC register by software. |
| (2) | Waits for an interrupt request. | Waits for detection of break field |
| (3) | | Detects a break field.<br>In the case of break field detection (LIN slave mode [fixed baud rate]). For details about the break field detection timing in the case of LIN slave mode [auto baud rate], see [Auto Baud Rate Correction Function]) |
| (4) | | • Detects a sync field ($55_H$)<br>• Baud rate generator setting (in the case of LIN slave mode [auto baud rate])<br>• Clears the no-response request bit (LNRR bit). |
| (5) | | • Receives an ID field.<br>• Checks an ID parity bit |
| (6) | | Sets a header reception complete flag. |
| (7) | • Checks the RLN3nLST register, and clears flags.<br>• Checks the RLN3nLIDB register, and prepares a response. | • Completes a header reception process.<br>• Waits for a response request. |

**NOTE**

The LIN/UART interface allows reception of break fields during frame transmission/reception. In that case, a framing error, bit error or other error may be detected at the stop bit position of the frame before the break field is received, and a status interrupt may occur as a result. For information about error detection, see **Section 18.5.3.7, Error Status**.

### [Auto Baud Rate Correction Function]

In LIN slave mode [auto baud rate], the system always measures the low-level widths that are received. If the first "Low level" width is 10 times (if the BLT bit of the RLN3nLBFC register is "0") or 11 times (if the BLT bit of the RLN3nLBFC register is "1") or greater calculated from the average of the starting 2 bits (the period of the consecutive fall edges from the beginning of the sync field) of the sync field, the system concludes that the detection of break field was successful, the system verifies that the data in the sync field is $55_H$. If the data in the sync field is indeed $55_H$ and the system judges that sync field reception was successful, the system automatically sets the baud rate correction result to the RLN3nLBRP01 register.

If data is received up to the ID field without error, a successful header reception interrupt is generated at the stop bit position.

On the other hand, if the data in the sync field is not $55_H$ and the system judges that sync field reception failed, the system sets the sync field error flag and an error interrupt is generated. In that case, baud rate correction is not performed and the LIN/UART interface waits for the detection of the next break field (low level).

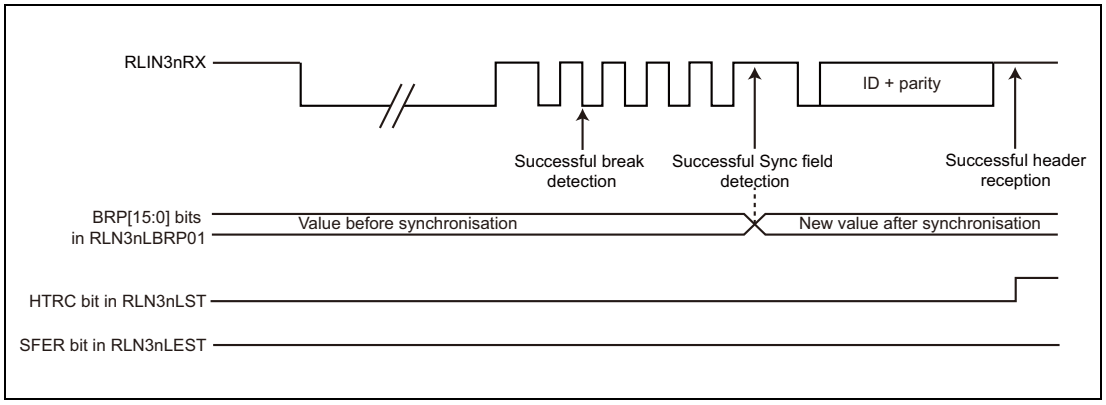



**Figure 18.8 Header Reception in LIN Slave Mode [Auto Baud Rate] (in Normal Operation)**

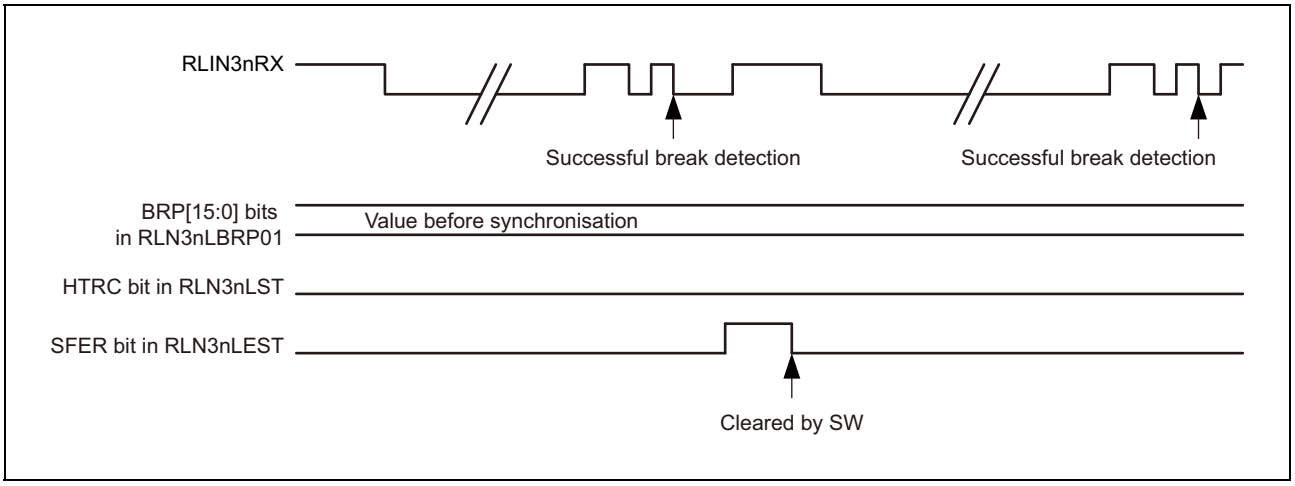



**Figure 18.9 Header Reception in LIN Slave Mode [Auto Baud Rate] (Sync Field Error)**

### (2) Response Transmission

**Figure 18.10** shows the operation of the LIN/UART interface (in LIN slave mode) in response transmission. **Table 18.79** provides processing in response transmission.

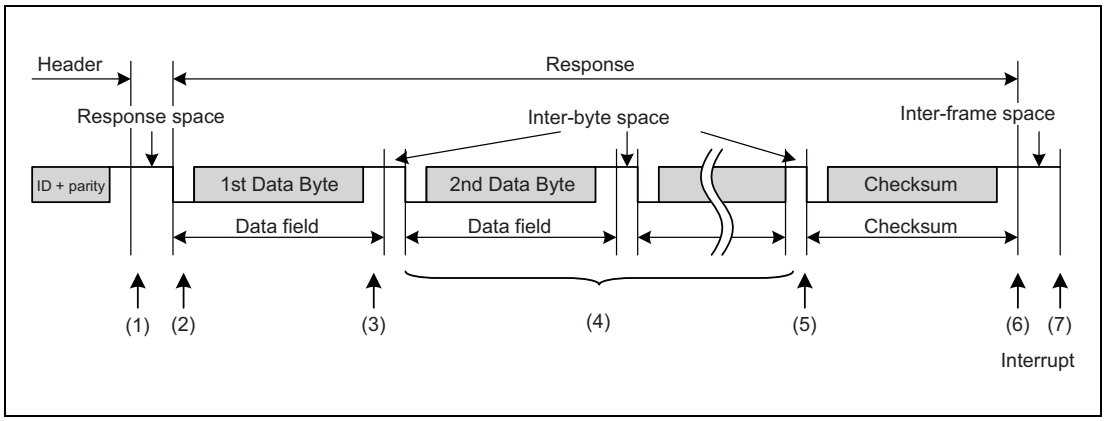



**Figure 18.10 Operation in Response Transmission**

**Table 18.79 Processing in Response Transmission (1/2)**

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (1) | • Sets the RLN3nLDFC register.<br>• Sets the RLN3nLDBRb registers. (b = 1 to 8)<br>• Sets the RTS bit in the RLN3nLTRC register to 1 (response transmission/ reception started) | • Waits for the setting of the RTS or LNRR bit of the RLN3nLTRC register by software<br>• Transmits the response space after the RTS bit of the RLN3nLTRC register is set to 1 |

**Table 18.79 Processing in Response Transmission (2/2)**

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (2) | Waits for an interrupt request. | Transmits 1st data byte. |
| (3) | | Transmits the inter-byte space. |
| (4) | | • Transmits 2nd data byte.<br>• Transmits an inter-byte space<br>• Transmits 3rd data byte.<br>• Transmits an inter-byte space<br>(Repeats the transmission of inter-byte spaces as many times as the data length specified in bits RFDL[3:0] in the RLN3nLDFC register, and stops the transmission when the BER bit in the RLN3nLEST register is 1 (bit error detected). If an error occurs, the checksum transmission in item (5) is not performed).<br>:<br>: |
| (5) | | Transmits the checksum. |
| (6) | | • Sets a successful frame/wake-up transmission flag.<br>• Sets the FTS bit in the RLN3nLTRC register to 0 (frame transmission or wake-up transmission/reception stopped)<br>[In frame separate mode]<br>• Sets the RTS bit in the RLN3nLTRC register to 0 (response transmission/reception stopped) |
| (7) | • Processing after communication<br>Checks the RLN3nLST register, and clears flags. | • Completes the response transmission process.<br>• Waits for a new break. |

**NOTE**

- For information about error detection, see **Section 18.5.3.7, Error Status**.
- The LIN/UART interface allows reception of break fields during frame transmission/reception. In that case, a framing error, bit error or other error may be detected at the stop bit position of the frame before the break field is received, and a status interrupt may occur as a result.

### (3) Response Reception

**Figure 18.11** shows the operation of the LIN/UART interface (LIN slave mode) in response reception. **Table 18.80** provides processing in response reception.

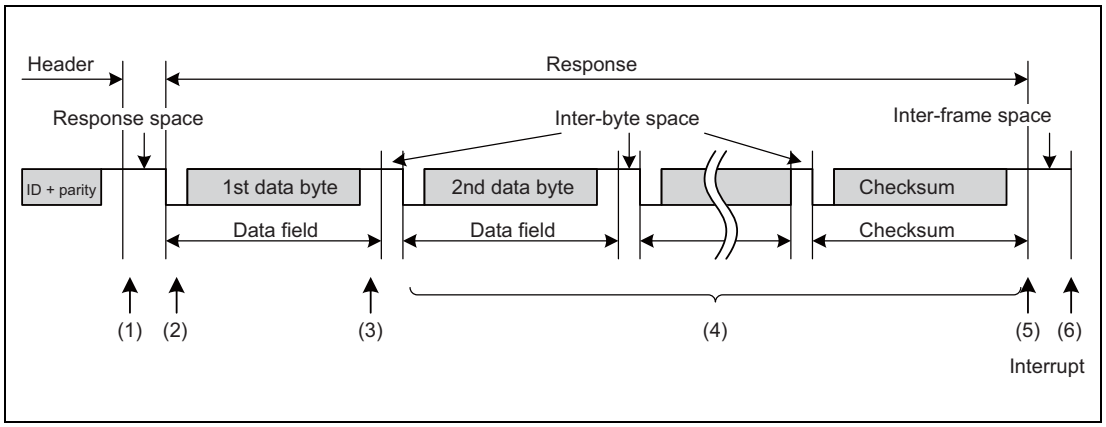



**Figure 18.11 Operation in Response Reception**

**Table 18.80 Processing in Response Reception**

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (1) | • Sets the RLN3nLDFC register.<br>• Sets the response transmission/ reception start bit (RTS bit) to 1. | • Waits for the setting by software of the response transmission/ reception start bit (RTS bit) or the no-response request bit (LNRR bit).<br>• Waits for detection of the start bit. |
| (2) | Waits for an interrupt request. | Receives 1st data byte when the start bit is detected. |
| (3) | | Sets the successful 1st data byte reception flag. |
| (4) | | • Receives 2nd data byte when the start bit is detected.<br>• Receives 3rd data byte when the start bit is detected.<br>Repeats the transmission of inter-byte spaces as many times as the data length specified in bits RFDL[3:0] in the RLN3nLDFC register, and stops the transmission when any bit in the RLN3nLEST register is 1 (bit error detected). If an error occurs, the checksum determination in item (5) is not performed).<br>:<br>:<br>• Receives the checksum when the start bit is detected. |
| (5) | | • Determines the checksum.<br>• Sets a successful frame/wake-up reception flag or an error flag.<br>• Sets the FTS bit in the RLN3nLTRC register to 0 (response transmission/reception stopped). |
| (6) | • Processing after communication Reads the received data. Checks the RLN3nLST register, and clears flags. | • Completes the response process.<br>• Waits for a new break. |

**NOTE**

- For information about error detection, see **Section 18.5.3.7, Error Status**.
- The LIN/UART interface allows reception of break fields during frame transmission/reception. In that case, a framing error, bit error or other error may be detected at the stop bit position of the frame before the break field is received, and a status interrupt may occur as a result.

### (4) No-Response Request

**Figure 18.12** shows the operation of the LIN/UART interface (LIN slave mode) when no response is requested. **Table 18.81** shows the processing that occurs when no response is requested.

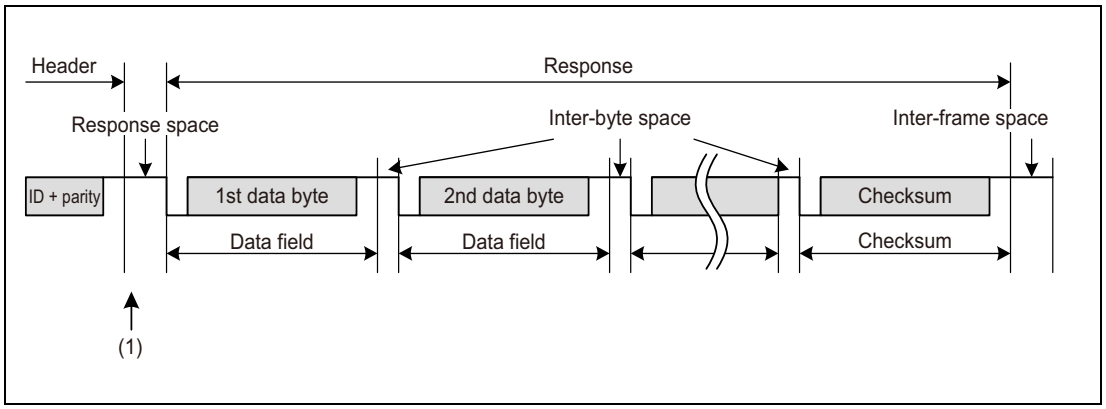



**Figure 18.12 Operation when No Response is Requested**

**Table 18.81 Processing when No Response is Requested**

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (1) | • Sets the no-response request bit (LNRR bit) to 1. | • Waits for setting of the no-response request bit (LNRR bit) by software<br>• Completes the frame reception process<br>• Waits for a new break |

### 18.5.3.3 Data Transmission/Reception

#### (1) Data Transmission

One bit of data is transmitted per 1 Tbit.

The data that is transmitted returns to the reception data input pin via the LIN transceiver. The received data and the transmitted data is compared bit by bit, and the results are stored in the BER flag in the RLN3nLEST register (see **Section 18.5.3.7, Error Status**).

In LIN master mode and LIN slave mode [fixed baud rate], 1 Tbit is generated to be 16fLIN, and thus the sampling point for received data is at the 13th clock cycle (81.25% position).

In LIN slave mode [auto baud rate], if 1 Tbit is generated to be 4fLIN, the sampling point for received data is at the third clock cycle (75% position). If 1 Tbit is generated to be 8fLIN, the sampling point for received data is at the 7th clock cycle (87.5% position).

**Figure 18.13** shows an example of data transmission timing.

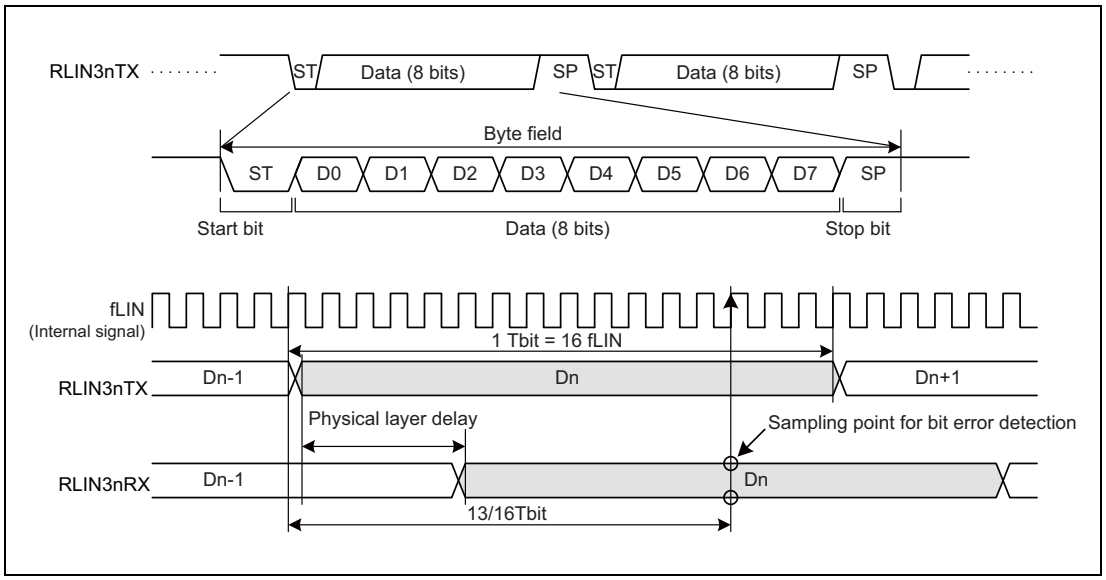



**Figure 18.13 Example of Data Transmission Timing (LIN Master Mode, LIN Slave Mode [Fixed Baud Rate])**

### (2) Data Reception

Data reception is performed by using the synchronized RLIN3nRX signal (an internal signal) that is the input from the RLIN3nRX pin synchronized with prescaler clock.

The byte field is synchronized at the falling edge of the start bit for the synchronized RLIN3nRX signal. After the falling edge is detected, sampling is performed again 0.5 Tbit later, and the falling edge is recognized as a start bit if the synchronized RLIN3nRX signal is low level. The falling edge is not recognized as a start bit if the RLIN3nRX signal after the clearing of the resetting is low-level-fixed or if a high level is detected on re-sampling.

After the start bit is detected, the system samples 1 bit per Tbit.

The LIN/UART interface has a noise filter function with respect to reception data. If the LRDNFS bit in the RLN3nLMD register is 0, the LIN/UART interface uses a noise filter, and for a sampling value the value determined by a 3-sampling majority rule on prescaler clocks is used. If the LRDNFS bit in the RLN3nLMD register is 1, the LIN/UART interface does not use a noise filter, and for a sampling value the value of the synchronized RLIN3nRX value at the sampling position is used as is.

**Figure 18.14** shows an example of data reception timing.

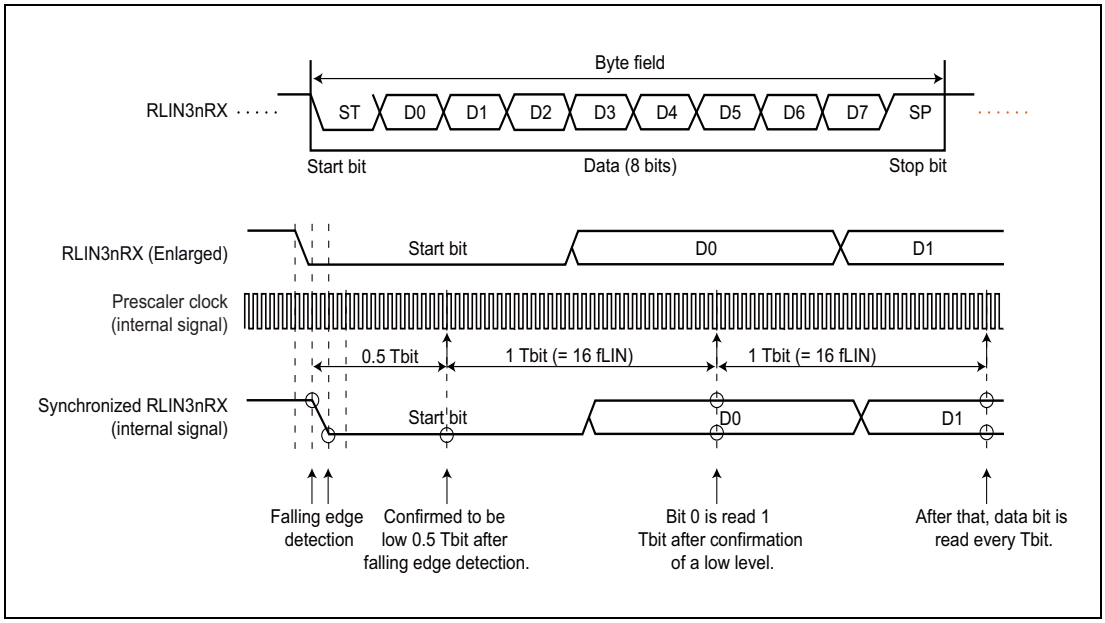



**Figure 18.14 Example of Data Reception Timing (LIN Master Mode, LIN Slave Mode [Fixed Baud Rate])**

### 18.5.3.4 Transmission/Reception Data Buffering

This section explains the buffer processing that takes place when the LIN/UART interface sends or receives data continuously.

#### (1) LIN Frame Transmission

For an 8-byte transmission, the contents stored in registers RLN3nLDBR1 to RLN3nLDBR8 are sequentially transmitted to data areas 1 to 8 of the LIN frame. In the case of a 4-bytes transmission, the contents stored in registers RLN3nLDBR1 to RLN3nLDBR4 are transmitted to data areas 1 to 4 of the LIN frame, but the contents of registers RLN3nLDBR5 to RLN3nLDBR8 are not transmitted. The transmitted checksum data is stored in the RLN3nLCBR register.

**Figure 18.15** depicts the LIN transmission processing and the required buffer.

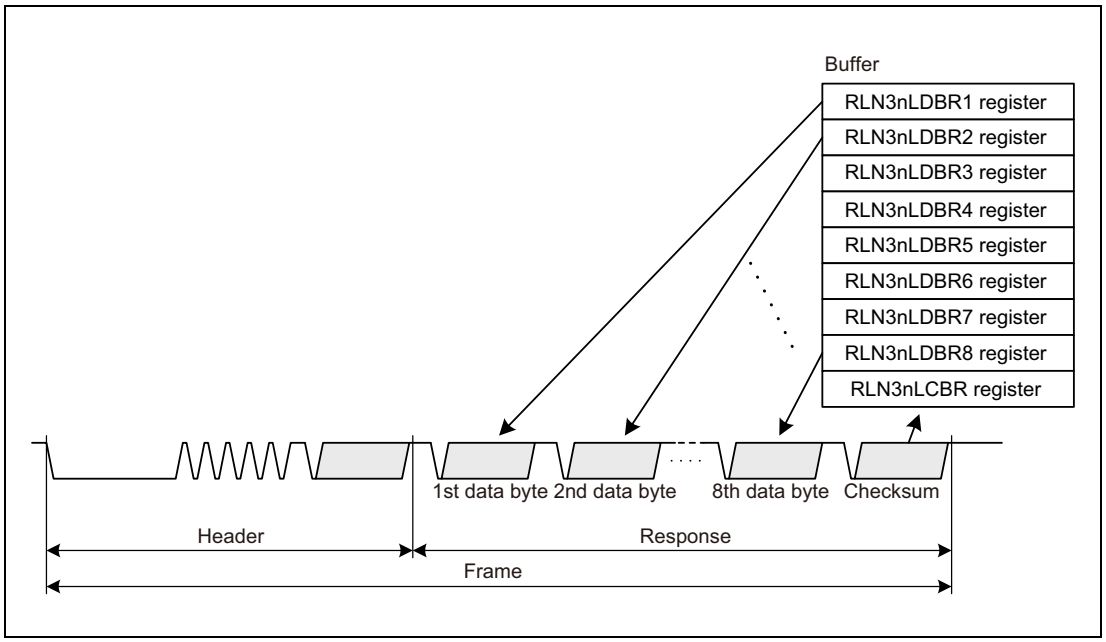



**Figure 18.15 LIN Transmission Processing and Required Buffer**

#### (2) Frame Separate Mode

Setting the FSM bit in the RLN3nLDFC register to 1 turns on the frame separate mode.

In frame separate mode, a header and a response are transmitted when prompted by separate transmission start requests.

When the transmission of a header is finished, the HTRC flag in the RLN3nLST register turns 1 (successful header transmission).

Use frame separate mode when sending or receiving response data of 9 bytes or greater in LIN master mode.

### (3) Reception of LIN Frames

For an 8-byte reception, the contents of data areas 1 to 8 of the LIN frame is stored in registers RLN3nLDBR1 to RLN3nLDBR8, respectively, upon receipt of a stop bit. In the case of a 4-byte reception, the contents of data areas 1 to 4 of the LIN frame are stored in registers RLN3nLDBR1 to RLN3nLDBR4, respectively; however, no data is stored in registers RLN3nLDBR5 to RLN3nLDBR8. Also, the received checksum data is stored in the RLN3nLCBR register.

**Figure 18.16** depicts the LIN reception processing and the required buffer.

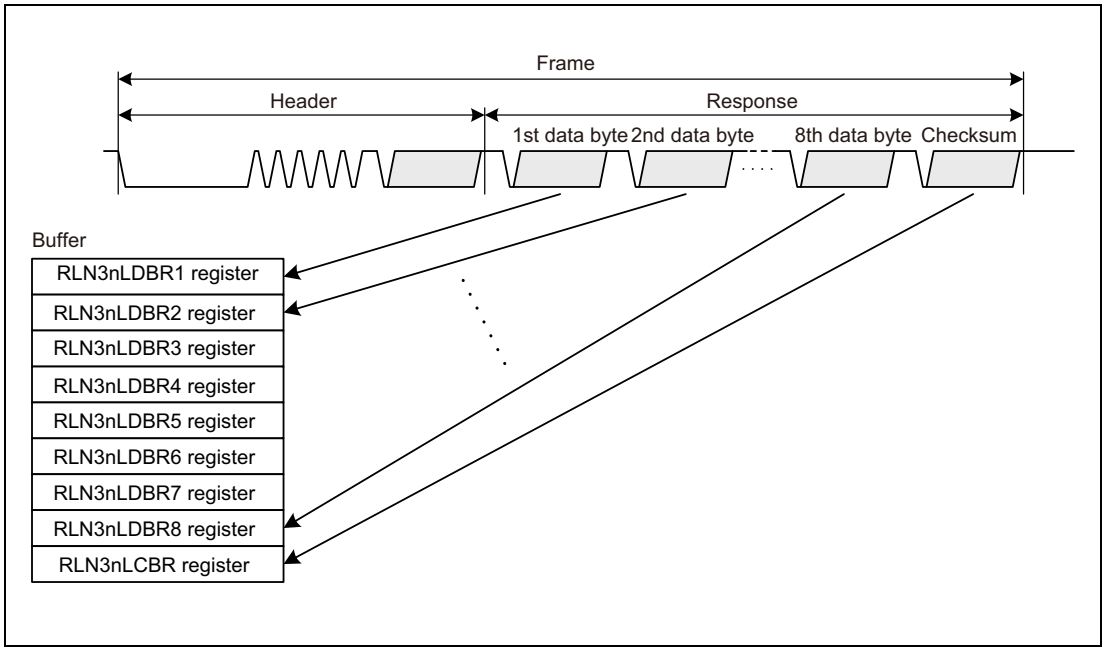



**Figure 18.16 LIN Reception Processing and Required Buffer**

### (4) Reception of 1st data byte

When the reception of the first byte of data is finished, the D1RC flag in the RLN3nLST register turns 1 (successful 1st data byte reception).

### (5) Multi-Byte Response Transmission/Reception Function

In normal LIN communication, a response is no more than or equal to 9 bytes (including the checksum field).In case responses are more than or equal to 9 bytes (plus checksum field) users can use response communication.

In such a case, the bit error, framing error, response preparation error detection, and auto checksum functions are enabled.

If the data length is greater than 8 bytes, the LSS bit in RLN3nLDFC register should be set to 1 (indicating that the next data group to be sent or received is not the final data group) in the first data group (variable in 0 to 8 bytes) before sending or receiving the data group. After the transmission or reception, the user should determine whether the next data group is the final data group. If it is the final data group, the LSS bit should be set to 0 (indicating that the next data group to be sent or received is the final data group), and a checksum should be appended to the final data group.

By changing the RFDL bit in RLN3nLDFC register settings when the RTS bit in RLN3nLTRC register is 0, the user can change the data length for each data group.

When performing multi-byte response transmission/reception in LIN master mode, set the FSM bit in RLN3nLDFC register in the RLN3nLDFC register to 1 (frame separate mode).

**NOTE**

In LIN slave mode, the LIN/UART interface can detect a new break field during the transmission or reception of a response.

#### 18.5.3.5 Wake-up Transmission/Reception

The wake-up transmission/reception can be used in LIN wake-up mode.

**(1) Wake-up Transmission**

In LIN wake-up mode, setting the RCDS bit in the RLN3nLDFC register to 1 (transmission) and the RFT bit in the RLN3nLDFC register to 1 (LIN master mode: response transmission), or setting the RCDS bit in the RLN3nLDFC register to 1 (LIN slave mode: response transmission) and the FTS bit in the RLN3nLTRC register to 1 (frame transmission, header reception or wake-up transmission/ reception started) causes a wake-up signal to be output from the output pin. The low width of the wake-up signal should be set using the WUTL[3:0] bits in the RLN3nLWUP register.

However, if the LWBR0 bit of the RLN3nLWBR register in LIN master mode is 1 (LIN2.x use), the LIN system clock (fLIN) becomes low level width at fa regardless of the setting of the LCKS bit of the RLN3nLMD register. By setting the WUTL[3:0] bit of the RLN3nLWUP register to $0100_B$ (5 Tbits), 260 µs low width can be output in LIN wake-up mode regardless of the setting of the LCKS bit of the RLN3nLMD register.

If a wake-up low is output without any bit error, the FTC flag in the RLN3nLST register turns 1 (successful frame response or wake-up transmission); when the FTCIE bit in the RLN3nLIE register is 1 (successful frame response/wakeup transmission interrupt enabled), an interrupt request is generated.

If a bit error is detected, wake-up transmission is canceled and the BER flag in the RLN3nLEST register is set to 1 (bit error detection).

**Figure 18.17** shows the wake-up transmission timing.

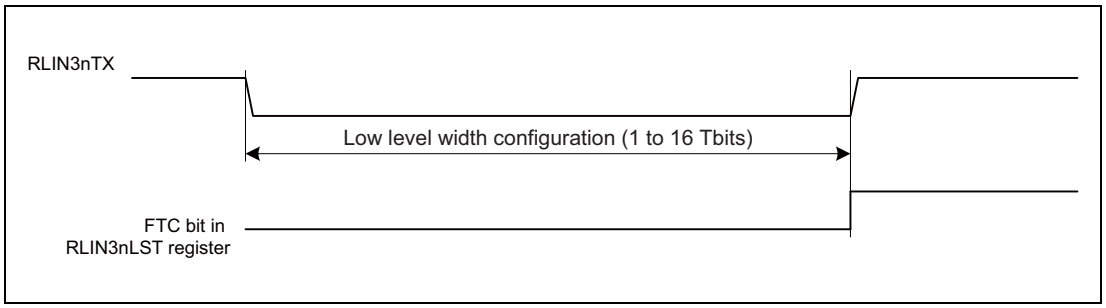



**Figure 18.17 Wake-up Transmission Timing**

### (2) Wake-up Reception

The detection of a wake-up involves the use of an input signal low level width count function.

The input signal low level width count function measures the low level width of the input signal to the RLIN3nRX pin, using the same sampling point as data reception. This allows the 2.5-Tbit or longer low-level width of the input signal of fLIN to be measured.

In LIN master mode, by setting the LWBR0 bit in the RLN3nLWBR register, operation is executed without changing the baud rate generator setting at a transition between LIN operation mode and LIN wake-up mode.

When LIN Specification Package Revision 1.3 is used, set the LWBR0 bit in the RLN3nLWBR register to 0. When LIN Specification Package Revision 2.x is used, set the LWBR0 bit to 1.

Setting the LWBR0 bit to 1 selects the LIN system clock (fLIN) to fa regardless of the setting of the LCKS bit in the RLN3nLMD register. (The LCKS bit is not changed). By setting the baud rate to 19200bps while fa is selected, the 130 µs or longer low-level width of the input signal to be measured regardless of the setting of the LCKS bit in the RLN3nLMD register.

When using this function, in LIN wake-up mode set the RFT bit in the RLN3nLDFC register to 0 (LIN master mode: response reception), the RCDS bit in the RLN3nLDFC register to 0 (LIN slave mode: response reception), or the FTS bit in the RLN3nLTRC register to 1 (frame transmission (header reception) or wake-up transmission/reception started).

When the low level width to be measured is reached, the FRC flag in the RLN3nLST register turns 1 (successful frame response/wake-up reception). If the FRCIE bit in the RLN3nLIE register is 1 (successful frame response or wake-up reception interrupt enabled), an interrupt request is generated.

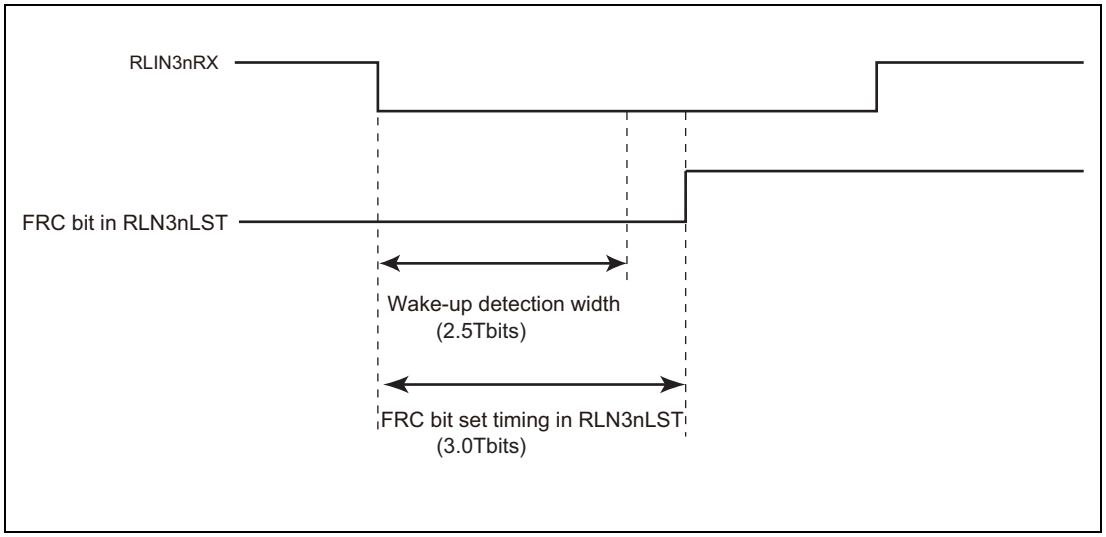



**Figure 18.18 Input Signal Low level Count Function**

### (3) Wakeup Collision

If the master node and the slave node transmit wakeup signals simultaneously, a collision will occur on the LIN bus, though a collision of wakeup signals is not detected in the LIN/UART interface.

### 18.5.3.6 Status

During LIN mode operation, the LIN/UART interface can detect seven types of statuses.

The four statuses, successful frame/wake-up transmission, successful frame/wake-up reception, error detection, successful header transmission/reception, can generate interrupt requests.

**Table 18.82** shows the types of statuses available in LIN master mode. **Table 18.83** lists the types of statuses available in LIN slave mode [auto baud rate] and in LIN slave mode [fixed baud rate].

**Table 18.82 Types of Statuses in LIN Master Mode**

| Status | Status set condition | Status clear condition | Operation mode capable of status detection | Corresponding bit | Interrupt |
|---|---|---|---|---|---|
| Reset | After the OM0 bit in the RLN3nLCUC register is set to not-LIN–reset-mode, if actually the LIN/UART interface is cleared from LIN reset mode. | After the OM0 bit in the RLN3nLCUC register is set to LIN reset mode, if actually the LIN/UART interface enters LIN reset mode. | All modes | OMM0 bit in RLN3nLMST register | — |
| Operation mode | After the OM1 bit in the RLN3nLCUC register is set to LIN operation mode, if actually the LIN/UART interface enters LIN operation mode. | After the OM1 bit in the RLN3nLCUC register is set to LIN wake-up mode, if actually the LIN/UART interface enters LIN wake-up mode. | • LIN operation mode<br>• LIN wake-up mode | OMM1 bit in RLN3nLMST register | — |
| Frame/wake-up transmission end | When a frame (header transmission + response transmission), a wake-up signal, or a data group is transmitted successfully. | • When another communication is started<br>• When cleared by software<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | FTC flag in RLN3nLST register | √ |
| Frame/wake-up reception end | When a frame (header transmission + response reception), a wake-up signal, or a data group is received successfully. | • When another communication is started<br>• When cleared by software<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | FRC flag in RLN3nLST register | √ |
| Error detection | If any of the PRER flag, CSER flag, FER flag, FTER flag, PBER flag, and BER flags in the RLN3nLEST register turns 1 (error detected). | • When another communication is started<br>• When cleared by software*1<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | ERR flag in RLN3nLST register | √ |
| 1st data byte reception end | The RFT bit in the RLN3nLDFC register is 0 (reception) and the first byte of the response field is received.*2 | • When another communication is started<br>• When cleared by software<br>• After transition to LIN reset mode | LIN operation mode | D1RC flag in RLN3nLST register | — |
| Header transmission end | When a header field is transmitted successfully. | • When another communication is started<br>• When cleared by software<br>• After transition to LIN reset mode | LIN operation mode | HTRC flag in RLN3nLST register | √ |

Note 1. In LIN operation mode, the ERR flag in the RLN3nLST register is cleared to 0 by writing 0 to the PRER flag, CSER flag, FER flag, FTER flag, PBER flag, or BER flags in the RLN3nLEST register.

Note 2. Not detected when the RFDL [3:0] bits in the RLN3nLDFC register are $0000_B$ (0-byte + checksum).

**Table 18.83 Types of Statuses in LIN Slave Mode**

| Status | Status set condition | Status clear condition | Operation mode capable of status detection | Corresponding bit | Interrupt |
|---|---|---|---|---|---|
| Reset | After the OM0 bit in the RLN3nLCUC register is set to not-LIN–reset-mode, if actually the LIN/UART interface is cleared from LIN reset mode. | After the OM0 bit of the RLN3nLCUC register is set to LIN reset mode, if actually the LIN/UART interface enters LIN reset mode. | All modes | OMM0 bit in RLN3nLMST register | — |
| Operation mode | After the OM1 bit in the RLN3nLCUC register is set to LIN operation mode, if actually the LIN/UART interface enters LIN operation mode. | • After the OM1 bit in the RLN3nLCUC register is set to LIN wake-up mode, if actually the LIN/UART interface enters LIN wake-up mode. | • LIN operation mode<br>• LIN wake-up mode | OMM1 bit in RLN3nLMST register | — |
| Frame/wake-up transmission end | When a response field, a wake-up signal, or a data group is transmitted successfully. | • When cleared by software<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | FTC flag in RLN3nLST register | √ |
| Frame/wake-up reception end | When a response field, a wake-up signal, or a data group is received successfully. | • When cleared by software<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | FRC flag in RLN3nLST register | √ |
| Error detection | If any of the PRER flag, IPER flag, CSER flag, SFER flag, FER flag, TER flag, and BER flag in the RLN3nLEST register turns 1 (error detected). | • When cleared by software*1<br>• After transition to LIN reset mode | • LIN operation mode<br>• LIN wake-up mode | ERR flag in RLN3nLST register | √ |
| 1st data byte reception end | The RCDS bit in the RLN3nLDFC register is 0 (reception) and the first byte of the response field is received.*2 | • When cleared by software<br>• After transition to LIN reset mode | LIN operation mode | D1RC flag in RLN3nLST register | — |
| Header reception end | When a header field is received successfully. | • When cleared by software<br>• After transition to LIN reset mode | LIN operation mode | HTRC flag in RLN3nLST register | √ |

Note 1. In LIN operation mode, the ERR flag in the RLN3nLST register is cleared to 0 by writing 0 to the PRER flag, IPER flag, CSER flag, SFER flag, FER flag, TER flag, or BER flag in the RLN3nLEST register.

Note 2. Not detected when the RFDL [3:0] bits in the RLN3nLDFC register are $0000_B$ (0-byte + checksum).

### 18.5.3.7 Error Status

#### (1) LIN Master Mode

##### (a) Types of Error Statuses

The LIN/UART interface can detect six types of error statuses in LIN master mode. The condition of these error statuses can be checked by means of the corresponding bits in the RLN3nLEST register.

All error statuses represent interrupt events.

**Table 18.84** shows the types of error statuses.

**Table 18.84 Types of Error Statuses in LIN Master Mode**

| Status | Error detection condition | Operation mode capable of error detection | Communication | Enable/ disable detection | Corresponding bit |
|---|---|---|---|---|---|
| Bit error | The transmitted data and the data on the LIN bus monitored by the receive pin do not match *1*2 | • LIN operation mode<br>• LIN wake-up mode | Cancel | Enabled | BER flag in RLN3nLEST register |
| Physical bus error | • LIN bus is detected to be high level when sending a break<br>• LIN bus is detected to be low level when sending a break delimiter<br>• LIN bus is detected to be high level when sending a wake-up | • LIN operation mode<br>• LIN wake-up mode | Cancel | Enabled | PBER flag in RLN3nLEST register |
| Timeout error | A frame or response transmission/reception does not terminate within a given time*3 | LIN operation mode | Cancel | Enabled | FTER flag in RLN3nLEST register |
| Framing error | In response field reception, a stop bit of each data byte is low level | LIN operation mode | Cancel | Enabled | FER flag in RLN3nLEST register |
| Checksum error | In response field reception, the result of checksum test gives an error | LIN operation mode | — | Disabled | CSER flag in RLN3nLEST register |
| Response preparation error | One of the following conditions occurs in frame combined mode during a multi-byte response reception:<br>• The first reception data byte is received after completion of header transmission but before a response transmission/reception request is set<br>• The first reception data byte is received after the completion of previous data group reception before a transmission/reception request for another data group is set. | LIN operation mode | Cancel | Disabled | RPER flag in RLN3nLEST register |

Note 1. If a bit error is detected, the process is canceled after a stop bit is sent. If a bit error is detected in a non-data area, such as an inter-byte space, the transmission is canceled immediately after that area. If a bit error is detected during the transmission of a wake-up, the transmission of the wake-up is canceled after the error-causing bit is sent.

Note 2. In a multi-byte response transmission, bit errors are detected also between data groups.

Note 3. The timeout time depends on the response field data length (the RFDL [3:0] bits in the RLN3nLDFC register) and the checksum selection (the CSM bit in the RLN3nLDFC register), and this can be calculated according to the following formula:
[Frame timeout]
On classic selection (when the CSM bit in RLN3nLDFC is 0): Timeout time = 49 + (number of data bytes + 1) × 14 [Tbit]
On enhanced selection (when the CSM bit in RLN3nLDFC is 1): Timeout time = 48 + (number of data bytes + 1) × 14 [Tbit]

The aforementioned timeout time is a time greater than the TFRAME_MAX of LIN Specification Package

Revision 1.3 on classic selection, or the TFRAME_MAX of LIN Specification Package Revision 2.x on enhanced selection.

[Response timeout]
Timeout time = (number of data bytes + 1) × 14 [Tbit]

The error status is cleared when the next communication is started, by software, or at a transition to LIN reset mode.

### (b) Target Time Area for LIN Error Detection

**Figure 18.19** shows the time domain in which the LIN/UART interface in LIN master mode performs monitoring for error detection.

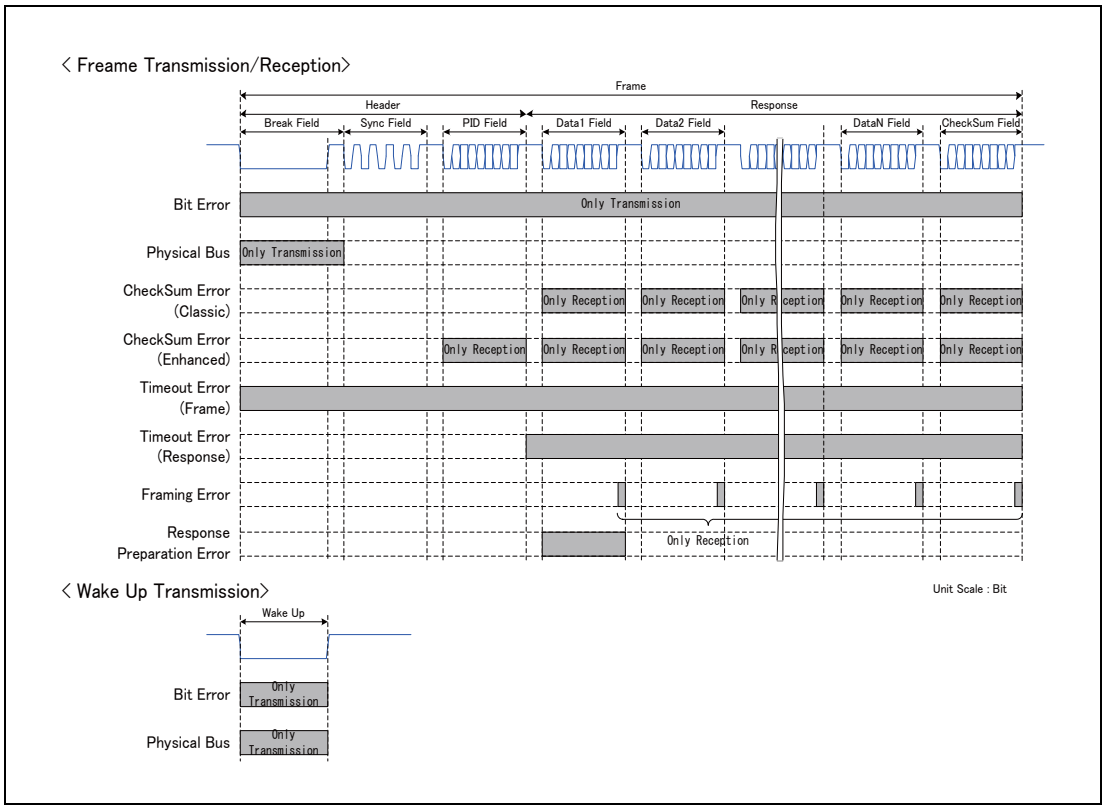



**Figure 18.19 Target Time Area for LIN Error Detection (LIN Master Mode)**

### (2) LIN Slave Mode

#### (a) Types of Error Statuses

The LIN/UART interface can detect seven types of error statuses in LIN slave mode [auto baud rate] or in LIN slave mode [fixed baud rate]. These error statuses can be verified by checking the corresponding bits in the RLN3nLEST register.

**Table 18.85** shows the types of error statuses.

**Table 18.85 Types of Error Statuses in LIN Slave Mode**

| Status | Error detection condition | Operation mode capable of error detection | Communication | Enable/ disable detection | Corresponding bit |
|---|---|---|---|---|---|
| Bit error | The transmitted data and the data on the LIN bus monitored by the receive pin do not match[*1] [*2] | • LIN operation mode<br>• LIN wake-up mode | Cancel | Enabled | BER flag in RLN3nLEST register |
| Timeout error | A frame or response transmission/reception does not terminate within a given time[*3] | LIN operation mode | Cancel | Enabled | TER flag in RLN3nLEST register |
| Framing error | In response field reception, a stop bit of each data byte is low level | LIN operation mode | Cancel | Enabled | FER flag in RLN3nLEST register |
| Sync field error | If the width of the break low level is greater than the width set by the LBLT bit in the RLN3nLBFC register and the sync field is not $55_H$ | LIN operation mode | Cancel | Enabled[*4] | SFER flag in RLN3nLEST register |
| Checksum error | In response field reception, the result of checksum test gives an error | LIN operation mode | —[*5] | Disabled | CSER flag in RLN3nLEST register |
| ID parity error | If the received ID parity bit does not match the value that is automatically calculated by the LIN/ UART interface | LIN operation mode | Cancel | Enabled | IPER flag in RLN3nLEST register |
| Response preparation error | • After the reception of a header, before the first reception data byte is received, response preparation is not made in time.<br>• Before the completion of receiving the first reception data byte for the next data group in a multi-byte response reception, response preparation for the next group is not made in time | LIN operation mode | Cancel | Disabled | RPER flag in RLN3nLEST register |

Note 1. If a bit error is detected, the process is canceled after a stop bit is sent. If a bit error is detected in a non-data area, such as an inter-byte space, the transmission is canceled immediately after that area. If a bit error is detected during the transmission of a wake-up, the transmission of the wake-up is canceled after the error-causing bit is sent.

Note 2. In a multi-byte response transmission, bit error can be detected between data groups.

Note 3. The timeout time depends on the response field data length (the RFDL [3:0] bits in the RLN3nLDFC register) and the checksum selection (the LCS bit in the RLN3nLDFC register), and this can be calculated according to the following formula. The time-out period until the RTS or LNRR bit of the RLN3nLTRC register is set is 8 data bytes. When the RTS bit is set, the timeout time is reset to the time based on the response field data length (RFDL[3:0] bit of the RLN3nLDFC register). When the LNRR bit is set, the timeout function stops.

[Frame timeout]
On classic selection (when the CSM bit in RLN3nLDFC is 0): Timeout time = 49 + (number of data bytes + 1) × 14 [Tbit]
On enhanced selection (when the CSM bit in RLN3nLDFC is 1): Timeout time = 48 + (number of data bytes + 1) × 14 [Tbit]

The aforementioned timeout time is a time greater than the TFRAME_MAX of LIN Specification Package Revision 1.3 on classic selection, or the TFRAME_MAX of LIN Specification Package Revision 2.x on enhanced selection.

[Response timeout]
Timeout time = (number of data bytes + 1) × 14 [Tbit]

Note 4. Only reflection of the result to the SFER flag can be enabled/disabled. Error detection cannot be enabled/ disabled.

Note 5. Checksum judgment is performed upon completion of response frame reception. In case of an error, the receive complete flag is not set to 1.

The error status is cleared by software or at a transition to LIN reset mode.

### (3) Target Time Area for LIN Error Detection

**Figure 18.20** shows the time domain in which the LIN/UART interface in slave mode performs monitoring for error detection.

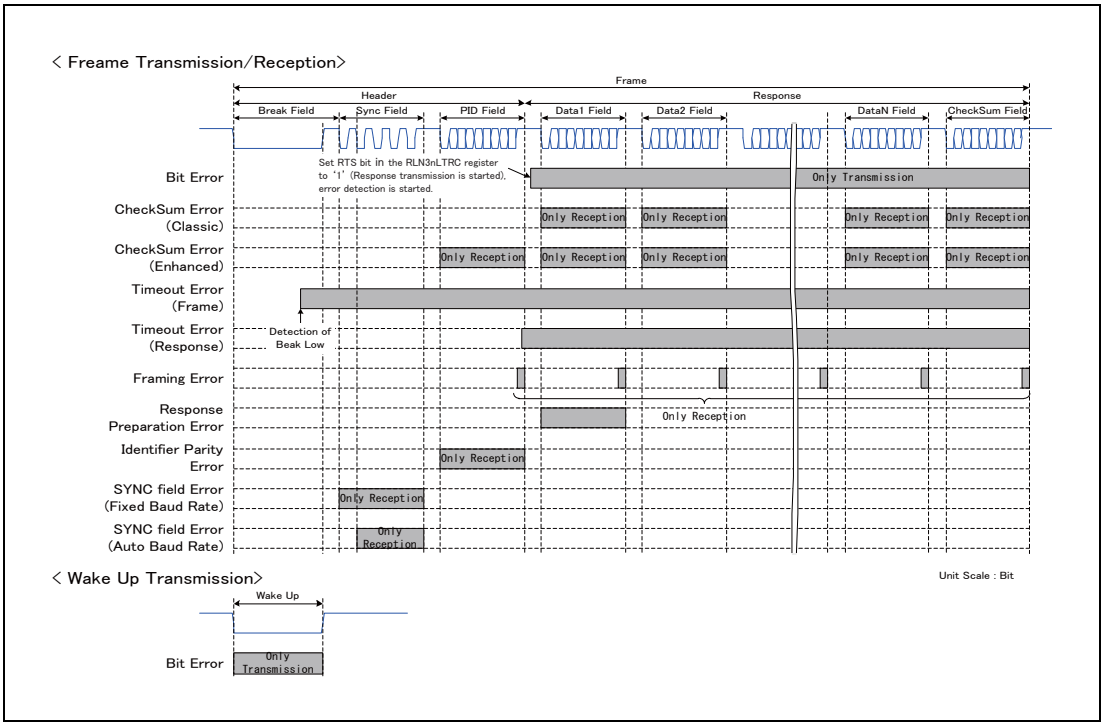



**Figure 18.20 Target Time Area for LIN Error Detection (LIN Slave Mode)**

## 18.5.4 UART Mode

In LIN reset mode, setting the LMD bits in the RLN3nLMD register to $01_B$ (UART mode) and the OM0 bit in the RLN3nLCUC register to 1 changes the mode to UART mode, turning the OMM0 bit in the RLN3nLMST register to 1.

### 18.5.4.1 Transmission

**Figure 18.21** shows LIN/UART interface (in UART mode) transmission operations; **Table 18.86** shows LIN/UART interface (in UART mode) transmission processing.

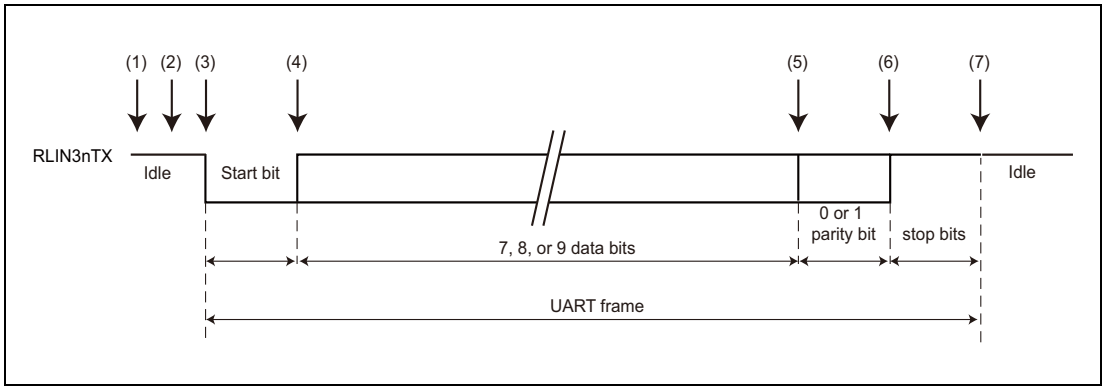



**Figure 18.21 LIN/UART Interface (in UART mode) Transmission Operation**

**Table 18.86 LIN/UART Interface (UART Mode) Transmission Processing (1/2)**

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (1) | • Sets a baud rate.<br>• Sets noise filter ON/OFF.<br>• Sets error detection enable.<br>• Sets data format.<br>• Sets an interrupt generation timing.<br>• Clears the LIN/UART interface from LIN reset mode.<br>• Sets the transmit enable bit (UTOE bit) to 1. | • Waits for a transmission trigger (RLN3nLUTDR register) by software. |
| (2) | • Sets the transmit data to the UART transmit data register (RLN3nLUTDR) or UART wait transmit data register (RLN3nLUWTDR). | • Sets the transmit status flag. |
| (3) | • Waits an interrupt request.<br>[When the UTIGTS bit is 0 (a transmission interrupt request is generated upon start of transmission)] | • Transmits a start bit (for switching between transmission and reception in half duplex communication, transmits a start bit after receiving 1 stop bit. For details about this function, see **(3) Transmission Start Wait Function**.<br>[When the UTIGTS bit is 0 (a transmission interrupt is generated)]<br>• Outputs a transmission interrupt. |
| (4) | • When transmitting data continuously, sets another piece of transmission data in the UART transmit data register (RLN3nLUTDR register), waits for the generation of an interrupt request. | Transmits the data set in the UART (for wait) transmit data register. |
| (5) | | Transmits a parity bit when parity is used. |
| (6) | | Transmits 1 or 2 stop bits. |

Table 18.86 LIN/UART Interface (UART Mode) Transmission Processing (2/2)

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (7) | [When the UTIGTS bit is 0 (a transmission interrupt request is generated upon start of transmission)]<br>• If another piece of transmission data is set, goes to step (3).<br><br>[When the UTIGTS bit is 1 (a transmission interrupt is output upon end of transmission)]<br>• When transmitting data continuously, goes to step (2). | [When the UTIGTS bit is 0 (a transmission interrupt request is generated upon start of transmission)]<br>• If another piece of transmission data is set, goes to step (3).<br>• If another piece of transmission data is not set, clears the transmit status flag.<br>[When the UTIGTS bit is 1 (a transmission interrupt is output upon end of transmission)]<br>• Outputs a transmission interrupt<br>• Clears the transmission status flag |

### (1) Continuous Transmission

The LIN/UART interface (in UART mode) can transmit multiple sets of data continuously by using the RLN3nLUTDR register. **Figure 18.22** shows an operation example where the transmission interrupt generation timing is the start of transmission and an operation example where the transmission interrupt generation timing is the end of transmission.

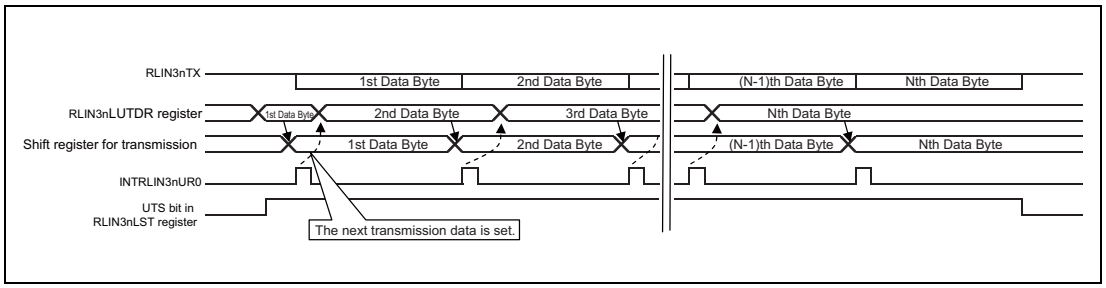



Figure 18.22 Operation Example of LIN/UART Interface (UART Mode) Continuous Transmission

An interrupt can be generated at the end of a transmission by changing the UTIGTS bit in the RLN3nLUOR1 register from 0 to 1 after the start of transmission of final data, provided only that the transmission interrupt generation timing is the start of transmission and the end of transmission of final data needs to be known.

### (2) UART Buffer Transmission

The LIN/UART interface (in UART mode) has a maximum of nine bytes of UART multi-byte data, and thus it is capable of performing continuous transmissions through the use of UART multi-byte data.

**Figure 18.23** shows the UART multi-byte data transmission operation in the LIN/UART interface (in UART mode). **Table 18.87** shows the UART multi-byte data transmission processing.

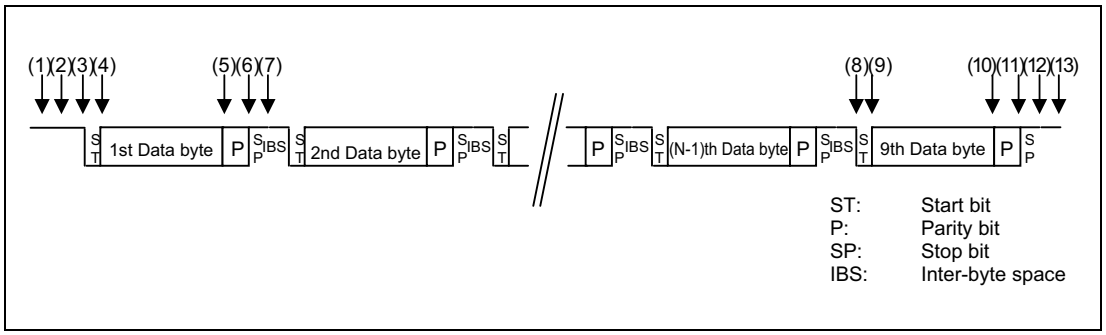



**Figure 18.23 UART Buffer Transmission in LIN/UART Interface (in UART mode for 9-byte transmission)**

**Table 18.87 UART Buffer Transmission Processing in LIN/UART Interface (in UART mode) (1/2)**

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (1) | • Sets a baud rate<br>• Sets noise filter ON/OFF<br>• Sets error detection enable<br>• Sets data format<br>• Sets an interrupt generation timing to the end of transmission.<br>• Clears the LIN/UART interface from LIN reset mode.<br>• Sets the transmit enable bit (UTOE bit) to 1 | • Waits for a transmission trigger (RTS bit) by software |
| (2) | • Sets the UART multi-byte data length and whether the system must wait for the start of transmission.<br>• Sets the transmission data in the UART data 0 buffer register (RLN3nLUDB0) and the LIN data buffer m register (RLN3nLDBRb). (b = 1 to 8)<br>• Sets the UART multi-byte data transmission start bit (RTS). | • Sets the transmit status flag. |
| (3) | Waits for an interrupt request. | Transmits a start bit.<br>(When switching from reception to transmission during half-duplex communication, transmits the start bit upon completion of the stop bit for reception. For details about this function, see **(3) Transmission Start Wait Function**.) |
| (4) | | Transmits the data set in the UART data buffer 0 register (RLN3nLUDB0) and the LIN/UART data buffer b register (RLN3nLDBRb). |
| (5) | | Transmits a parity bit when parity is used. |
| (6) | | Transmits 1 or 2 stop bits |
| (7) | | Transmits an inter-byte space (idle). |
| | | Repeats steps (3) to (7) until frame count -1 that was set in the UART multi-byte data length select bits is reached. |

**Table 18.87 UART Buffer Transmission Processing in LIN/UART Interface (in UART mode) (2/2)**

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (8) | | Transmits a start bit. |
| (9) | | Transmits the data set in the LIN/UART data buffer b register (RLN3nLDBRb). |
| (10) | | Transmits a parity bit when parity is used. |
| (11) | | Transmits 1 or 2 stop bits. |
| (12) | | • Sets the buffer transmission end flag.<br>• Clears the UART multi-byte data transmit start bit (RTS).<br>• A transmission interrupt request signal.<br>• Clears the transmission status flag. |
| (13) | • Checks the RLN3nLST register, and clears flags<br>• In the case of continuous data transmission, goes to step (2). | |

### (1) UART Buffer Transmission

For a 9-byte transmission, the contents stored in the RLN3nLUDB0 and RLN3nLDBR1 to RLN3nLDBR8 registers are transmitted to data areas 1 to 9. The RLN3nLUDB0 register is used only if 9-byte transmission is set. In other cases, the RLN3nLDBR1 to RLN3nLDBR8 registers are selected depending upon the length of data involved. For a 4-byte transmission, the contents stored in the RLN3nLDBR1 to RLN3nLDBR4 registers are transmitted to data areas 1 to 4, but the contents of the RLN3nLDBR5 to RLN3nLDBR8 registers are not transmitted. An RLIN3n transmission interrupt is generated after the transmission of the data that is set in the MDL [3:0] bits of the RLN3nLDFC register. The spaces between transmission data items can be set in the IBS bit in the RLN3nLSC register.

**Figure 18.24** shows a UART multi-byte data and the transmission processing.

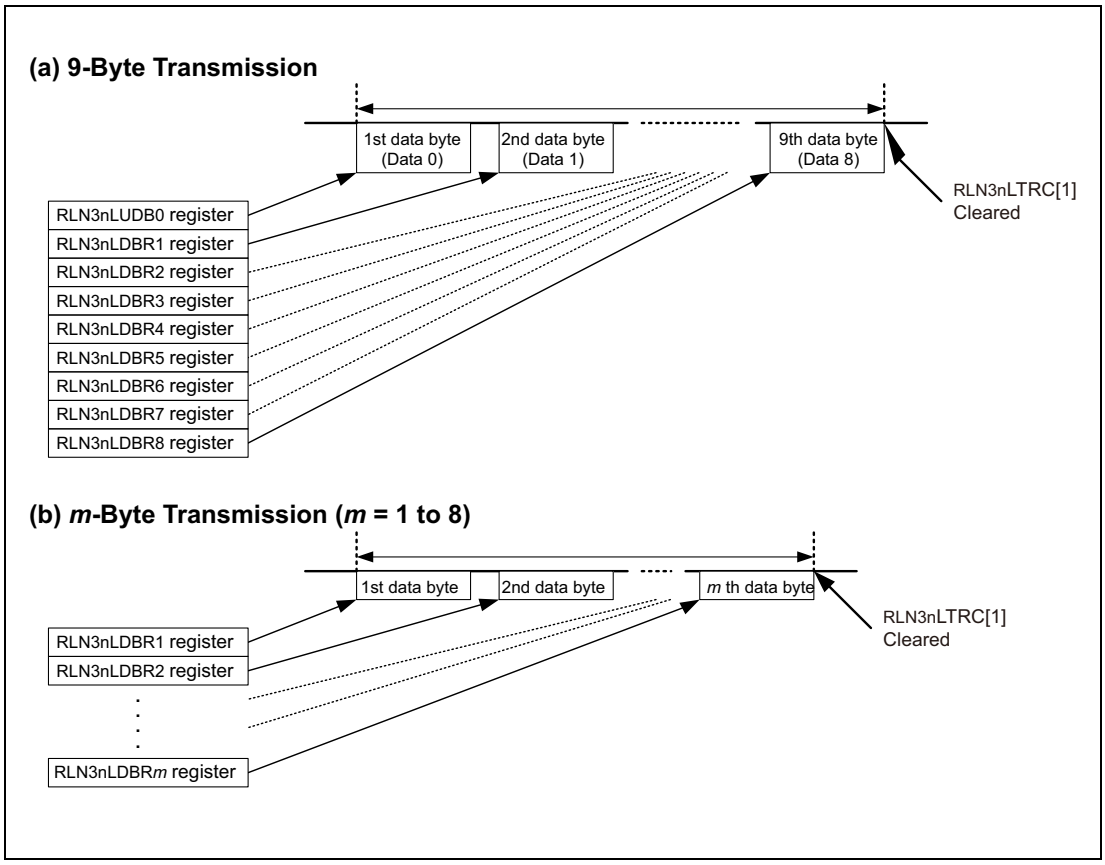



**Figure 18.24 UART Buffer and Transmission Processing**

### (2) Data Transmission

One bit of data is transmitted per Tbit.

In single-wire (half-duplex) communication, if the BERE bit in the RLN3nLEDE register is 1 (bit error detection enabled), the transmission data and the input pin level are compared bit by bit during data transmission, and the results are stored in the BER flag in the RLN3nLEST register (see **Section 18.5.3.7, Error Status**). The timing at which the input pin is sampled during data transmission can vary depending upon the settings of the LPRS[2:0] and NSPB[3:0] bits in the RLN3nLWBR register.

The bit error detection timing in UART mode is shown in **Table 18.88**.

**Table 18.88 Error Detection Timing in UART Mode**

| Sampling count per bit | Bit error detection timing |
|---|---|
| 6 samples | 3rd clock cycle + 1 prescaler clock |
| 7 samples | 4th clock cycle + 1 prescaler clock |
| 8 samples | 4th clock cycle + 1 prescaler clock |
| 9 samples | 5th clock cycle + 1 prescaler clock |
| 10 samples | 5th clock cycle + 1 prescaler clock |
| 11 samples | 6th clock cycle + 1 prescaler clock |
| 12 samples | 6th clock cycle + 1 prescaler clock |
| 13 samples | 7th clock cycle + 1 prescaler clock |
| 14 samples | 7th clock cycle + 1 prescaler clock |
| 15 samples | 8th clock cycle + 1 prescaler clock |
| 16 samples | 8th clock cycle + 1 prescaler clock |

Example of Data Transmission Timing (when 1 Tbit = 16 Sampling) is shown in **Figure 18.25**

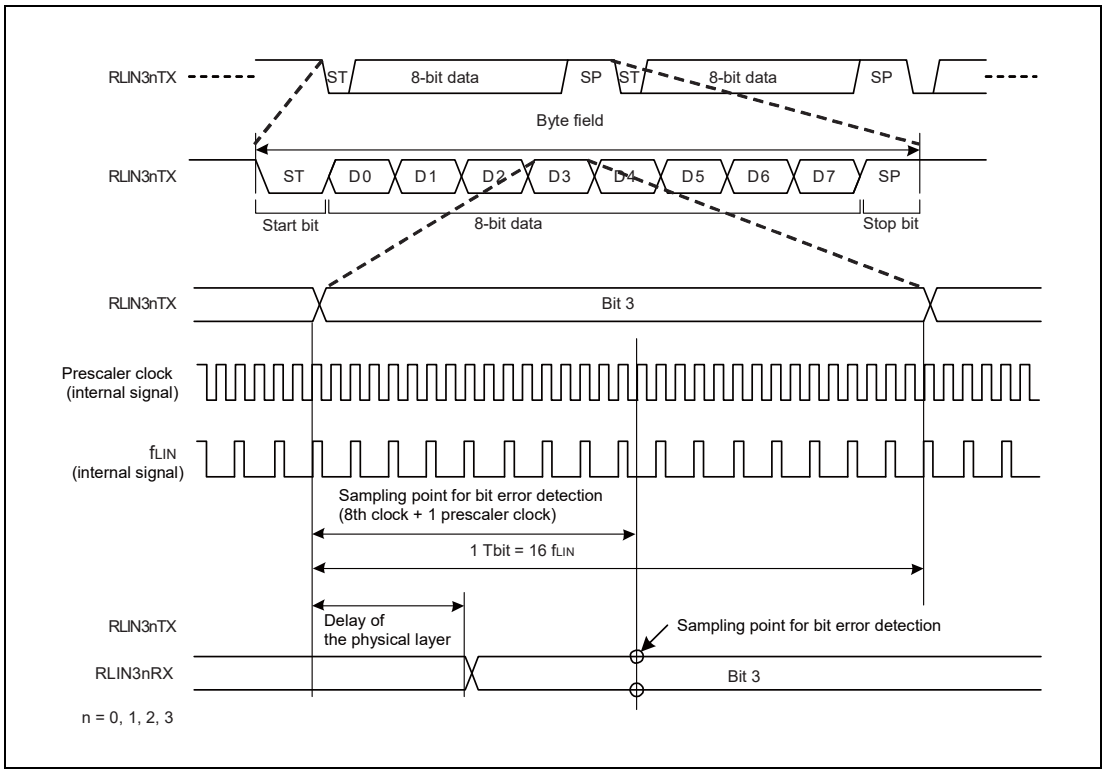



**Figure 18.25 Example of Data Transmission Timing (When 1 Tbit = 16 Sampling)**

### (3) Transmission Start Wait Function

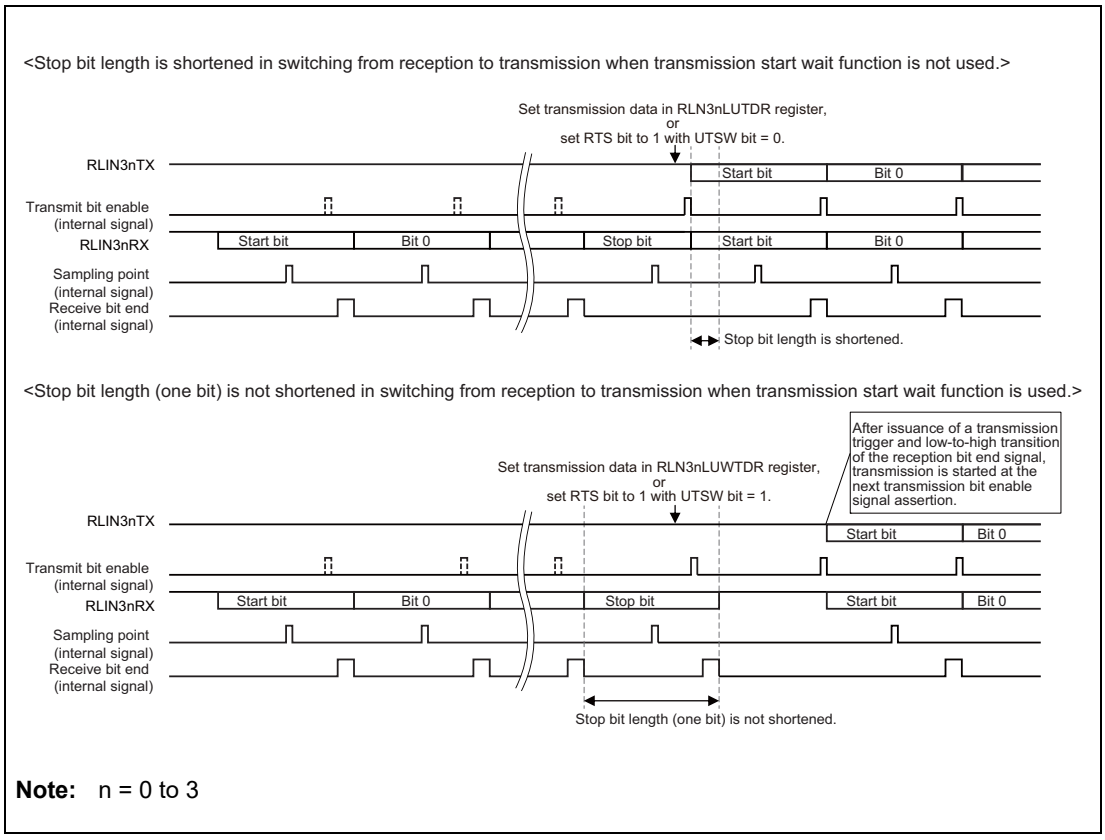
For performing half-duplex communication, the LIN/UART interface (in UART mode) has the function of securing the reception stop bit length when switching from reception to transmission.

If it is desired to delay the start of transmission until the stop bits for the reception are completed, set data in the RLN3nLUWTDR register, which is used only for the wait function, instead of setting transmission data in the RLN3nLUTDR register as a start-of-transmission request. When transmitting from the UART multi-byte data, set 1 (UART multi-byte data transmission started) in the RTS bit in the RLN3nLTRC register with 1 set in the UTSW bit in the RLN3nLDFC register.

In such a case, the LIN/UART interface delays the start of transmission until the stop bits of reception data are completed.

It should be noted that even if the UART stop bit length select bit (USBLS) in RLN3nBLFC register is 1 (stop bits = 2 bits), delay is made only for 1 bits.

**Figure 18.26** shows the operation of transmission wait function.




**Note:** n = 0 to 3

**Figure 18.26 Transmission Wait Function (If Transmission Data is Set during the Stop Bits in the Received Data)**

### 18.5.4.2 Reception

**Figure 18.23** shows the LIN/UART interface (in UART mode) reception operation**Table 18.89**. shows the LIN/UART interface (in UART mode) reception processing.

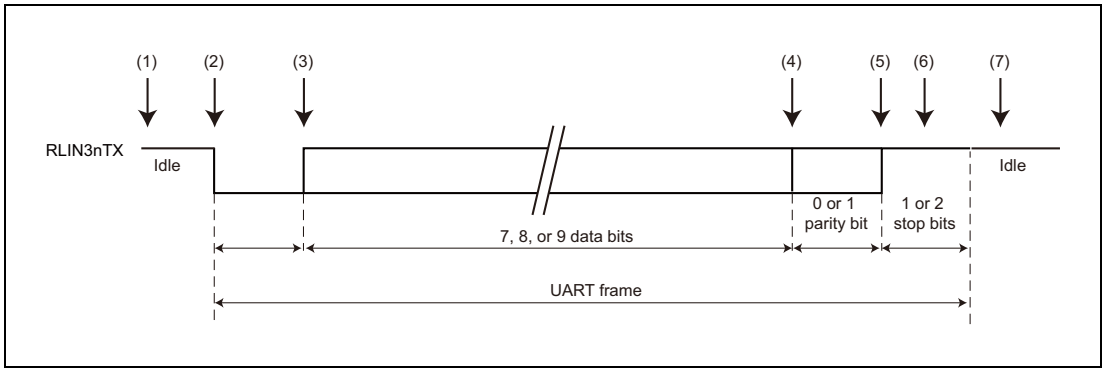



**Figure 18.27 LIN/UART Interface (in UART Mode) Reception Operation**

**Table 18.89 LIN/UART Interface (in UART Mode) Reception Processing**

| | Software processing | LIN/UART interface processing |
|---|---|---|
| (1) | • Sets a baud rate.<br>• Sets noise filter ON/OFF.<br>• Sets error detection enable.<br>• Sets data format.<br>• Clears the LIN/UART interface from LIN reset mode.<br>• Sets the receive enable bit (UROE bit) to 1. | • Waits for reception enable state switching by software.<br>• Waits for detection of a start bit. |
| (2) | Waits for an interrupt request. | • Waits for a falling edge from the reception pin, and detects a start bit.<br>• Sets the reception status flag. |
| (3) | | Receives data. |
| (4) | | Receives a parity bit when parity is used. |
| (5) | | Receives only 1 stop bit. |
| (6) | | • A successful reception interrupt request signal<br>• Clears the reception status flag. |
| (7) | Checks the RLN3nLST register, and clears flags | Waits for a falling edge from the reception pin. |

### (1) Data Reception

Data reception is performed by using the synchronized RLIN3nRX (an internal signal) that is the input from the RLIN3nRX pin synchronized with the prescaler clock.

The byte field is synchronized at the falling edge of the start bit for the synchronized RLIN3nRX signal.

After the falling edge is detected, resampling is performed 0.5 Tbits later when the number of the sampling per 1 Tbit is even and {(the number of the sampling + 1) / 2} / (the number of sampling) Tbits later when the number is odd. If the synchronized RLIN3nRX signal is low level, the bit is recognized as a start bit.

The bit is not recognized as a start bit if the RLIN3nRX signal is fixed at low level after the reset is cleared or if a high level is detected during the resampling.

After the start bit is detected, 1 bit is sampled per Tbit.

However, when the BERE bit in the RLN3nLEDE register is 1, the sampling point is the same as the bit error detection timing.

The LIN/UART interface has a noise filter function with respect to received data. If the LRDNFS bit in the RLN3nLMD register is 0, the noise filter is used. For a sampling value, the value determined by a 3-sampling majority rule by the prescaler clock is used. If the LRDNFS bit in the RLN3nLMD register is 1, the noise filter is not used. In this case, for a sampling value, the synchronized RLIN3nRX value at the sampling position is used as is.

**Figure 18.28** shows an example of data reception timing.

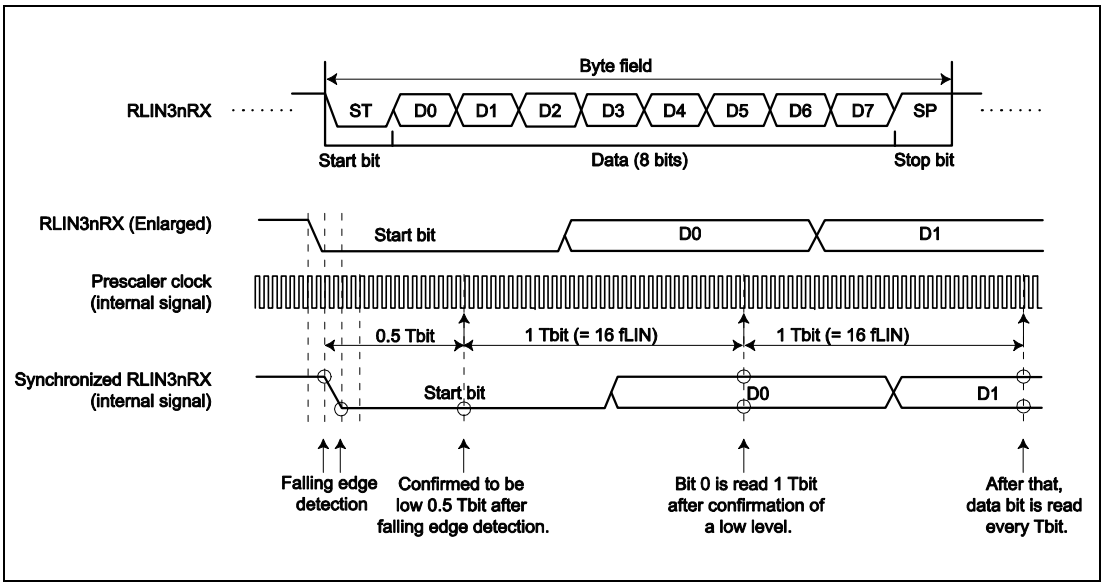



**Figure 18.28 Example of Data Reception Timing (When Sampling Count is 16 in 1 Tbit)**

### 18.5.4.3 Expansion Bits

The LIN/UART interface (in UART mode) can transmit and receive 9-bit long data by setting the UEBE bit in the RLN3nLUOR1 register to 1.

#### (1) Expansion Bit Transmission

The LIN/UART interface (in UART mode) can transmit 9-bit long data when the expansion bit enable bit (UEBE) in the UART option register 1 (RLN3nLUOR1) is 1 and by writing the 9-bit data to either the UART transmission data register (RLN3nLUTDR) or the UART wait transmission data register (RLN3nLUWTDR).

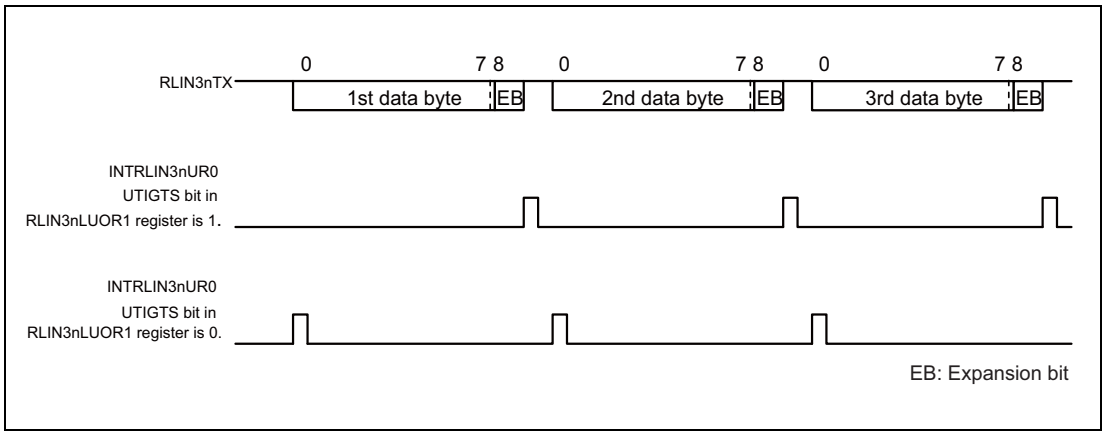



**Figure 18.29 Transmission Example When Expansion Bit is Enabled (LSB First)**

#### (2) Expansion Bit Reception

With the LIN/UART interface (in UART mode), 9-bit data can always be received without requiring a comparison of expansion bits, provided that the expansion bit enable bit (UEBE) in the UART option register 1 (RLN3nLUOR1) is 1, the expansion bit comparison disable bit (UECD) is 1, and the expansion bit data comparison enable bit (UEBDCE) is 0. Irrespective of the particular setting of the expansion bit detection level select bit (UEBDL) in the UART option register 1 (RLN3nLUOR1), a successful RLIN3n reception interrupt is generated (n = 0 to 3) when 9-bit data is received.

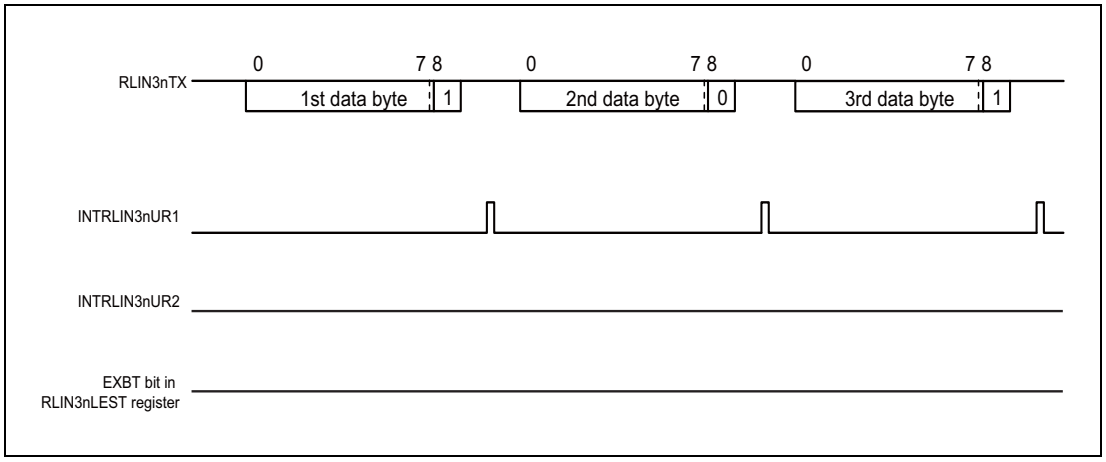



**Figure 18.30 Expansion Bit Reception Example (LSB First)**

### (3) Expansion Bit Reception (with Expansion Bit Comparison)

The LIN/UART interface (in UART mode) can compare received expansion bits and the UEBDL bits when the expansion bit enable bit (UEBE) in the UART option register 1 (RLN3nLUOR1) is 1 and the expansion bit comparison disable bit (UECD) is 0 and the expansion bit/data comparison enable bit (UEBDCE) is 0.

If the level that was set in the expansion bit detection level select bit (UEBDL) is detected, an RLIN3n status interrupt is generated upon completion of data reception, and the expansion bit detection flag (EXBT) in the LIN error status register (RLN3nLEST) is set. If the reversed value of an expansion bit detection level is detected, a successful RLIN3n reception interrupt is generated. In either case, the received data is stored in the UART reception data register (RLN3nLURDR), unless there was an overrun error.

RLIN3n Module does not judge the expansion bit comparison, if the reception error (parity error, framing error) occurs. RLIN3n Module judge the expansion bit comparison, if the reception error (overrun error) occurs.

**Figure 18.31** shows an example when the expansion bit detection level select bit (UEBDL) is set to 0.

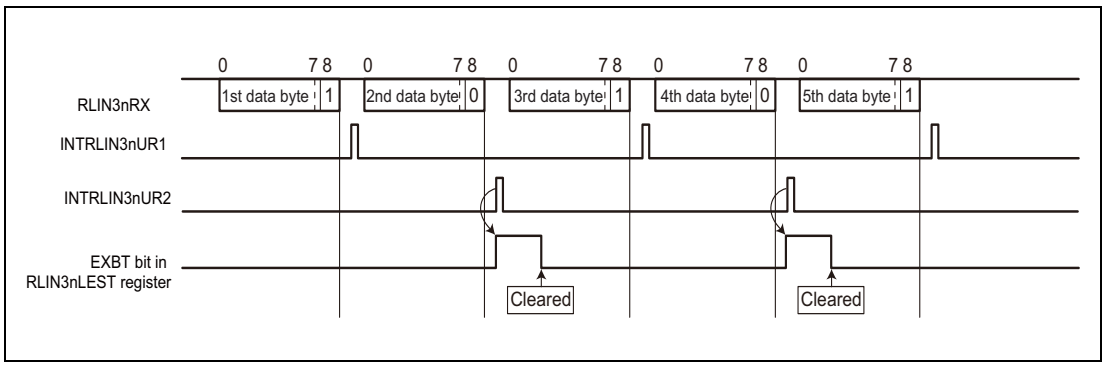



**Figure 18.31 Expansion Bit Reception Example (with Expansion Bit Comparison) (LSB First, UEBDL = 0)**

**NOTE**

- If a reception error (parity error, framing error, or overrun error) occurs in received data 0, 2, or 4 (if a reversed value of an expansion bit detection level is detected), an RLIN3n status interrupt is generated, and the error flag is updated. In this case, a successful RLIN3n reception interrupt is not generated.
- If a reception error (parity error, framing error, or overrun error) occurs in received 1st data byte or 3 (if an expansion bit detection level is detected), an RLIN3n status interrupt is generated, and the error flag is updated. If the overrun error occurs, the expansion bit detection flag (EXBT) is also set.

### (4) Expansion Bit Reception (with Data Comparison)

If the expansion bit enable bit (UEBE) in the UART option register 1 (RLN3nLUOR1) is 1 and the expansion bit comparison disable bit (UECD) is 0 and the expansion bit/data comparison enable bit (UEBDCE) is 1, and if the level that was set by the expansion bit detection level select bit (RLN3nUEBDL) is detected, the LIN/UART interface compares the 8 bits, exclusive of the expansion bit in the received data, with the a pre-set RLN3nLIDB register value.

If the result of the comparison is a match, the LIN/UART interface performs the following operations:

- Generates an RLIN3n reception status interrupt
- Sets an expansion bit detection flag (EXBT)
- Sets an ID match flag (IDMT)
- Stores the received data in the UART reception data register (RLN3nLURDR)

Even when the result of the comparison is a match, a successful RLIN3n reception interrupt is not generated.

If the result of the comparison is not a match, no successful RLIN3n reception interrupt or RLIN3n status interrupt is generated, and the EXBT and IDMT flags are not set to 1. The received data is stored in the UART reception data register (RLN3nLURDR).

When changing the UEBDCE bit to 0, make the change before the reception of another set of data is finished.

**Figure 18.32** shows an example when the expansion bit detection level select bit (UEBDL) is set to 0.

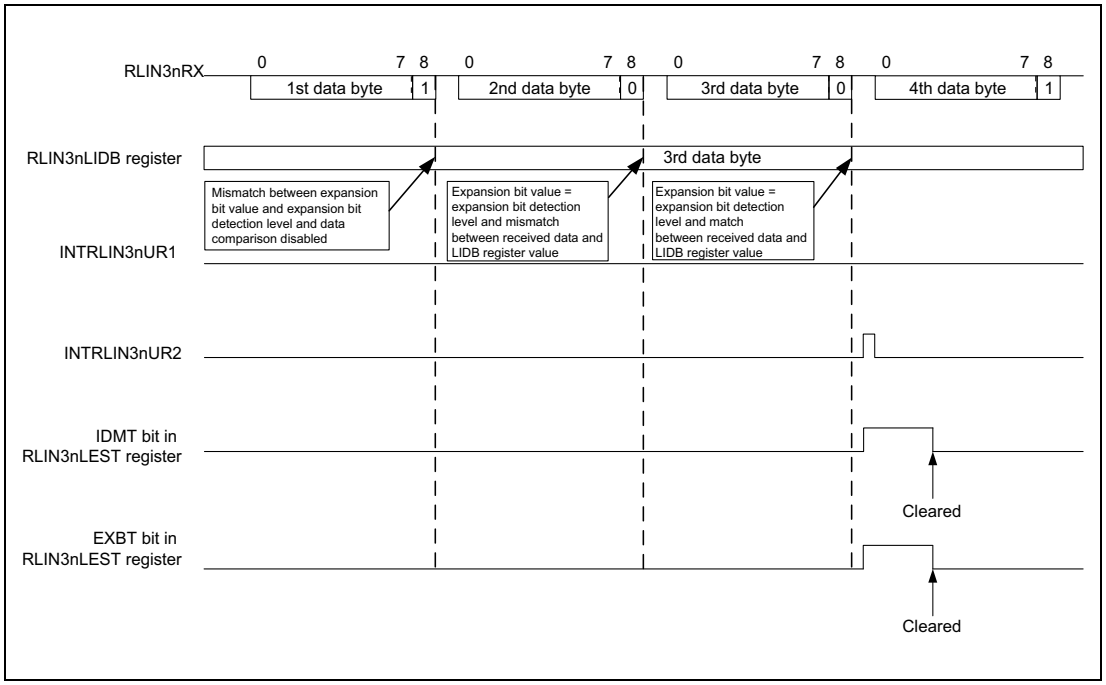



**Figure 18.32 Expansion Bit Reception Example (with Data Comparison) (LSB First, UEBDL = 0)**

### 18.5.4.4 Status

In UART mode, the LIN/UART interface can detect five types of statuses.

Two statuses, successful UART multi-byte data transmission and error detection, can generate interrupt requests.

**Table 18.90** shows the types of statuses available in UART mode.

**Table 18.90 Types of Statuses in UART Mode**

| Status | Status set condition | Status clear condition | Corresponding bit | Interrupt |
|---|---|---|---|---|
| Reset | After the OM0 bit in the RLN3nLCUC register is set to not-LIN–reset-mode, if actually the LIN/UART interface is cleared from LIN reset mode. | After the OM0 bit in the RLN3nLCUC register is set to LIN reset mode, if actually the LIN/UART interface enters LIN reset mode. | OMM0 bit in RLN3nLMST register | — |
| Successful UART multi-byte data transmission | When the transmission is finished of data equal to the length set in the MDL bits in the RLN3nLDFC register<br>• When the UTIGTS bit in the RLN3nLUOR1 register is 0 (transmission interrupt request is generated upon start of transmission), the transmission of the last data of the data length set by the MDL bit in the RLN3nLDFC register is started.<br>• When the UTIGTS bit in the RLN3nLUOR1 register is 1 (transmission interrupt request is generated upon end of transmission), the transmission of the data length set by the MDL bit in the RLN3nLDFC register is ended. | • When cleared by software<br>• After transition to LIN reset mode | FTC flag in RLN3nLST register | √ |
| Error detection | If any of the UPER flag, IDMT flag, EXBT flag, FER flag, OER flag, and BER flags in the RLN3nLEST register turns 1 (error detected). | • When cleared by software*1<br>• After transition to LIN reset mode | ERR flag in RLN3nLST register | √ |
| Transmission status | • When data is written to the RLN3nLUTDR or RLN3nLUWTDR register.<br>• When a 1 is written to the RTS bit in the RLN3nLTRC register. | • The transmission of the data set in the RLN3nLUTDR or RLN3nLUWTDR register is complete, but another transmission data item is not set<br>• The transmission of the data in the UART multi-byte data is complete, and the RTS bit in the RLN3nLTRC register is cleared<br>• After transition to LIN reset mode | UTS flag in RLN3nLST register | — |
| Reception status | • When a start bit is detected. | • When a sampling point for stop bits is detected<br>• After transition to LIN reset mode | URS flag in RLN3nLST register | — |

Note 1. Writing a 0 to the UPER, IDMT, EXBT, FER, OER, and BER flags in the RLN3nLEST register when the LIN reset mode is being canceled turns the ERR flag in the RLN3nLST register to 0.

### 18.5.4.5 Error Status

#### Types of Error Statuses

In UART mode, the LIN/UART interface can detect four types of errors and two types of statuses. The condition of these statuses can be checked by means of the corresponding bits in the RLN3nLEST register.

**Table 18.91** lists applicable status types.

**Table 18.91 Types of Statuses in UART Mode**

| Status | Error detection condition | Communication | Enable/ disable detection | Corresponding bit |
|---|---|---|---|---|
| Bit error | The transmitted data and the data monitored on the receive pin do not match*1 | Continues until the transmission of the set transmission data is finished. | Enabled | BER flag in RLN3nLEST register |
| Overrun error | After received data is stored in the RLN3nLURDR register, another data item is received before the data is read. (In this case, no data is stored in the RLN3nLURDR register). | — (Reception is finished by the time this error is detected) | Enabled | OER flag in RLN3nLEST register |
| Framing error | When the first stop bit is low level in the reception processing. | — (Reception is finished by the time this error is detected) | Enabled | FER flag in RLN3nLEST register |
| Parity error | The received parity value fails to match the parity value calculated from the received data | Continues until the data reception is finished. | Disabled *2 | UPER flag in RLN3nLEST register |
| Expansion bit detection | The value of the received expansion bit matches the value of the UEBDL bit in the RLN3nLUOR1 register. | — | Enabled | EXBT flag in RLN3nLEST register |
| ID match detection | The value of the received expansion bit matches the value of the UEBDL bit in the RLN3nLUOR1 register and the 8-bit receive data excluding the expansion bit matches the value of the RLN3nLIDB register. | — | Enabled | IDMT flag in RLN3nLEST register |

Note 1. In the case of transmission from the UART multi-byte data, bit errors are detected even in the space between UART frames (interval space).

Note 2. Setting the UPS[1:0] bits in the RLN3nLBFC register to $10_B$ (0 parity) disables the checking of parity bit values. In this case, no parity error is generated.

The error status is cleared by software or at a transition to LIN reset mode.

### 18.5.5 LIN Self-Test Mode

When the LIN/UART interface enters the LIN self-test mode, RLIN3nTX and RLIN3nRX are disconnected from external pins and RLIN3nTX and RLIN3nRX are connected in the LIN/UART interface. Therefore, the frame transmitted from RLIN3nTX is looped back to RLIN3nRX.

The LIN self-test mode can perform tests exclusively in LIN mode.

The self-test can be performed in the following four types.

- LIN master self-test mode (transmission): Header transmission and response transmission
- LIN master self-test mode (reception): Header transmission and response reception
- LIN slave self-test mode (transmission): Header reception and response transmission
- LIN slave self-test mode (reception): Header reception and response reception

In LIN self-test mode, the operate is at the fastest baud rate, regardless of the setting of the baud rate generator.

Regardless of the setting of the baud rate related registers, the baud rate operates at the LIN communication clock source/16 [bps]. (The NSPB bits in the RLN3nLWBR register should be set to $0000_B$ or $1111_B$.)

In addition, in LIN self-test mode, the following functions are not supported.

- LIN wake-up mode
- Frame separate mode
- Multi-byte response transmission/reception
- LIN slave mode (Auto baud rate)
- Frame/response timeout error

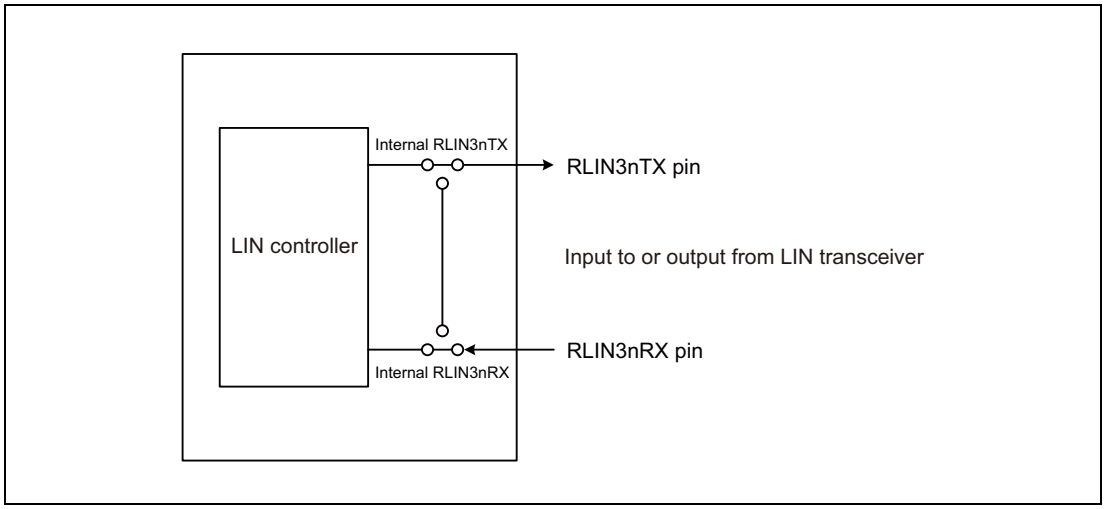



**Figure 18.33 Connection in LIN Reset Mode, LIN Mode, and UART Mode**

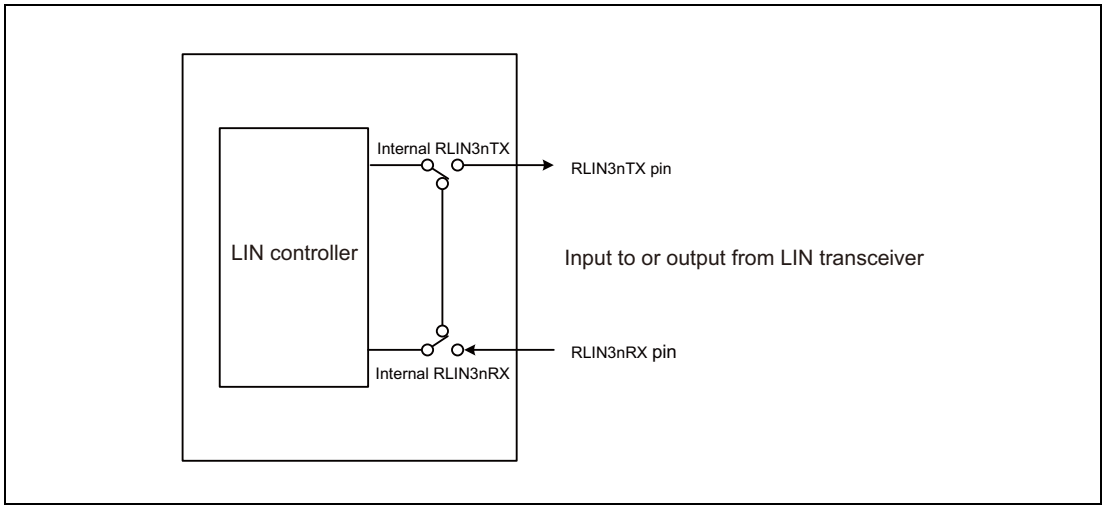



**Figure 18.34 Connection in LIN Self-Test Mode**

#### 18.5.5.1 Change to LIN Self-Test Mode

Writing to the RLN3nLSTC register makes a transition to the LIN self-test mode.
When the LSTM bit in the RLN3nLSTC register is set to 1, the shift to the LIN self-test mode is checked.

When changing to LIN self-test mode, be sure to execute a specific sequence. In that sequence, information must be written three times consecutively to the LIN self-test control register, as follows:

- Change to LIN reset mode
  Set the OM0 bit in the RLN3nLCUC register to 0 (LIN reset mode).
  Read the OMM0 bit in the RLN3nLMST register; verify that it is 0 (LIN reset mode).
- Select a LIN mode
  LMD bits in RLN3nLMD = $00_B$ (LIN master mode) or $11_B$ (LIN slave mode [fixed baud rate])
- 1st write: RLN3nLSTC register = 1010 0111 ($A7_H$)
- 2nd write: RLN3nLSTC register = 0101 1000 ($58_H$)
- 3rd write: RLN3nLSTC register = 0000 $0001_B$ ($01_H$)
- Verify the transition to LIN self-test mode
  Read the LSTM bit in the RLN3nLSTC register; verify that it is 1 (LIN self-test mode).

If the key of the first write ($A7_H$) is written twice by mistake, the transition to LIN self-test mode is canceled. The above sequence should be retried from the step of first write. In addition, if a write to another LIN-related register is performed during transition to LIN self-test mode (three consecutive write operations to the RLN3nLSTC register), the transition is also canceled.

### 18.5.5.2 Transmission in LIN Master Self-Test Mode

To execute a self-test on LIN master transmission, perform the procedure below:

- Set the baud rate, noise filter, and interrupt output related registers.
  RLN3nLWBR register = $0000xxxx_B$ *1
  RLN3nLBRP0 register = $xxxxxxxx_B$ *1
  RLN3nLBRP1 register = $xxxxxxxx_B$ *1
  RLN3nLMD register = $00xxxx00_B$ *1
- Set the interrupt enable and error enable related registers.
  RLN3nLIE register = $0000xxxx_B$ *2
  RLN3nLEDE register = x000x0xx
- Set the break field and space related registers.
  RLN3nLBFC register = $00xxxxxx_B$
  RLN3nLSC register = $00xx0xxx_B$
- Cancel the LIN reset mode.
  Write $11_B$ to the OM1 and OM0 bits in the RLN3nLCUC register, and check that the OMM1 and OMM0 bits in the RLN3nLMST register are $11_B$.
- Set the transmit frame related registers.
  RLN3nLDFC register = $00x1xxxx_B$
  RLN3nLIDB register = $xxxxxxxx_B$
  RLN3nLDRB1 to RLN3nLDRB8 registers = $xxxxxxxx_B$
- Header transmission → response transmission started
  Set the FTS bit in the RLN3nLTRC register to 1 (frame transmission or wake-up transmission/ reception started).
  The LIN master self-test mode (transmission) is executed. In this mode, interrupt are generated, and status and error status are also updated. The checksum is automatically calculated by the LIN/ UART interface.
- When the transmission is completed, the reversed value of the looped-back frame data is stored in the RLN3nLIDB, RLN3nLDBRb (b = 1 to 8), and RLN3nLCBR registers (the data is reversed before being stored because the transmitted value should be compared with the looped-back value). Then, the FTS bit in the RLN3nLTRC register is cleared.
- If the transmission fails to complete due to an error, the applicable error flag is set and the FTS bit in the RLN3nLTRC register is cleared.

**Note:** x: Don't care

**Note 1.** The following register settings are not reflected to the operation of the LIN self-test mode. The LPRS bit in the RLN3nLWBR register, the RLN3nLBRP0 register, the RLN3nLBRP1 register and the LCKS bit in the RLN3nLMD register. Therefore, those settings are not necessary.

**Note 2.** If necessary, set the related registers described in **Section 6, Interrupt Controller (INTC)**.

**Note 3.** When the successful header transmission interrupt and the successful frame transmission interrupt are used in the same interrupt processing, if the software processing of the successful header transmission interrupt is not completed before the generation of the successful frame transmission interrupt, the SHIE bit in the RLN3nLIE register should not be set to 1 (successful header transmission interrupt enabled). The time required from the set of the successful header transmission flag to the set of the successful frame/wake-up transmission flag is calculated by the following formula.

10 × (number of data bytes + 1) [Tbit]
1 Tbit = frequency of LIN communication clock source × 16

#### 18.5.5.3 Reception in LIN Master Self-Test Mode

To execute a self-test on LIN master reception, perform the procedure below:

- Set the baud rate, noise filter, and interrupt output related registers.
  RLN3nLWBR register = $0000xxxx_B$ *1
  RLN3nLBRP0 register = $xxxxxxxx_B$ *1
  RLN3nLBRP1 register = $xxxxxxxx_B$ *1
  RLN3nLMD register = $00xxxx00_B$ *1
- Set the interrupt enable and error enable related registers.
  RLN3nLIE register = $0000xxxx_B$ *2
  RLN3nLEDE register = $x000x0xx_B$
- Set the break field and space related registers.
  RLN3nLBFC register = $00xxxxxx_B$
  RLN3nLSC register = $00xx0xxx_B$ *1
- Cancel the LIN reset mode.
  Write $11_B$ to the OM1 and OM0 bits in the RLN3nLCUC register, and check that the OMM1 and OMM0 bits in the RLN3nLMST register are $11_B$.
- Set the reception frame related registers.
  RLN3nLDFC register = $00x0xxxx_B$
  RLN3nLIDB register = $xxxxxxxx_B$
  RLN3nLDRB1 to RLN3nLDRB8 registers = $xxxxxxxx_B$
  RLN3nCBR register = $xxxxxxxx_B$
  Since the checksum value to be transmitted is not automatically calculated, set the calculation value to the RLN3nLCBR register.
- Header transmission → response reception started
  Set the FTS bit in the RLN3nLTRC register to 1 (frame transmission or wake-up transmission/ reception started).
  The LIN master self-test mode (reception) is executed. In this mode, interrupt are generated, and status and error status are also updated.
- When the reception is completed, the reversed value of the looped-back frame data is stored in the RLN3nLIDB, RLN3nLDBRb (b = 1 to 8), and RLN3nLCBR registers (the data is reversed before being stored because the set value should be compared with the looped-back value). Then, the FTS bit in the RLN3nLTRC register is cleared.
- If the reception fails to complete due to an error, the applicable error flag is set and the FTS bit in the RLN3nLTRC register is cleared.

**Note:** x: Don't care

**Note 1.** The following register settings are not reflected to the operation of the LIN self-test mode. The LPRS bit in the RLN3nLWBR register, the RLN3nLBRP0 register, the RLN3nLBRP1 register, the LCKS bit in the RLN3nLMD register, and the IBS bit in the RLN3nLSC register. Therefore, those settings are not necessary.

**Note 2.** If necessary, set the related registers described in **Section 6, Interrupt Controller (INTC)**.

**Note 3.** When the successful header transmission interrupt and the successful frame reception interrupt are used in the same interrupt processing, if the software processing of the successful header transmission interrupt is not completed before the generation of the successful frame reception interrupt, the SHIE bit in the RLN3nLIE register should not be set to 1 (successful header transmission interrupt enabled).
The time required from the set of the successful header transmission flag to the set of the successful frame/wake-up reception flag is calculated by the following formula.

10 × (number of data bytes + 1) [Tbit]

1 Tbit = 1/LIN communication clock source × 16

#### 18.5.5.4 Transmission in LIN Slave Self-Test Mode

To execute a self-test on LIN slave transmission, perform the procedure below:

- Set the baud rate, noise filter, and interrupt output related registers.
  RLN3nLWBR register = $0000xxx0_B$ *1
  RLN3nLBRP0 register = $xxxxxxxx_B$ *1
  RLN3nLBRP1 register = $xxxxxxxx_B$ *1
  RLN3nLMD register = $00xx0011_B$
- Set the interrupt enable and error enable related registers.
  RLN3nLIE register = $0000xxxx_B$ *2
  RLN3nLEDE register = $xx0xx00x_B$
- Set the break field and space related registers.
  RLN3nLBFC register = $0000000x_B$ *3
  RLN3nLSC register = $00xx0001_B$
- Cancel the LIN reset mode.
  Write $11_B$ to the OM1 and OM0 bits in the RLN3nLCUC register, and check that the OMM1 and OMM0 bits in the RLN3nLMST register are $11_B$.
- Set the transmit frame related registers.
  RLN3nLDFC register = $00x1xxxx_B$
  RLN3nLIDB register = $xxxxxxxx_B$
  RLN3nLDBR1 to RLN3nLDBR8 registers = $xxxxxxxx_B$
- Header reception → response transmission started
  Set the FTS bit in the RLN3nLTRC register to 1 (header reception or wake-up transmission/ reception started).
  (Without any setting of the RTS bit in the RLN3nLTRC register, the header reception and the response transmission are executed in this order.)
  The LIN slave self mode (transmission) is executed. In this mode, interrupt are generated, and status and error status are also updated.
  The checksum is automatically calculated by the LIN/UART interface.
- When the transmission is completed, the reversed value of the looped-back frame data is stored in the RLN3nLIDB, RLN3nLDBRb (b = 1 to 8), and RLN3nLCBR registers (the data is reversed before being stored because the transmitted value should be compared with the looped-back value). Then, the FTS bit in the RLN3nLTRC register is cleared.
- If the transmission fails to complete due to an error, the applicable error flag is set and the FTS bit in the RLN3nLTRC register is cleared.

**Note:** x: Don't care

**Note 1.** The following register settings are not reflected to the operation of the LIN self-test mode. The LPRS bit in the RLN3nLWBR register, the RLN3nLBRP0 register, and the RLN3nLBRP1 register. Therefore, those settings are not necessary.

**Note 2.** If necessary, set the related registers described in **Section 6, Interrupt Controller (INTC)**

**Note 3.** According to the setting of this register, 9.5-Tbit or 10.5-Tbit width break is output from the internal RLIN3nTX.

**Note 4.** When the successful header reception interrupt and the successful response transmission interrupt are used in the same interrupt processing, if the software processing of the successful header reception interrupt is not completed before the generation of the successful response transmission interrupt, the SHIE bit in the RLN3nLIE register should not be set to 1 (successful header reception interrupt enabled). The time required from the set of the successful header reception flag to the set of the successful response/wake-up reception flag is calculated by the following formula.

10 × (number of data bytes + 1) [Tbit]

1 Tbit = frequency of 1/LIN communication clock source × 16

#### 18.5.5.5 Reception in LIN Slave Self-Test Mode

To execute a self-test on LIN slave reception, perform the procedure below:

- Set the baud rate, noise filter, and interrupt output related registers.
  RLN3nLWBR register = $0000xxx0_B$ *1
  RLN3nLBRP0 register = $xxxxxxxx_B$ *1
  RLN3nLBRP1 register = $xxxxxxxx_B$ *1
  RLN3nLMD register = $00xx0011_B$
- Set the interrupt enable and error enable related registers.
  RLN3nLIE register = $0000xxxx_B$ *2
  RLN3nLEDE register = $xx0xx00x_B$
- Set the break field and space related registers.
  RLN3nLBFC register = $0000000x_B$ *3
  RLN3nLSC register = $00xx0001_B$ *1
- Cancel the LIN reset mode.
  Write $11_B$ to the OM1 and OM0 bits in the RLN3nLCUC register, and check that the OMM1 and OMM0 bits in the RLN3nLMST register are $11_B$.
- Set the reception frame related registers.
  RLN3nLDFC register = $00x0xxxx_B$
  RLN3nLIDB register = $xxxxxxxx_B$
  RLN3nLDBR1 to RLN3nLDBR8 registers = $xxxxxxxx_B$
  RLN3nCBR register = $xxxxxxxx_B$
  Since the checksum value to be transmitted is not automatically calculated, set the calculation value to the RLN3nLCBR register.
- Header reception → response reception started
  Set the FTS bit in the RLN3nLTRC register to 1 (header reception or wake-up transmission/ reception started).
  (Without any setting of the RTS bit in the RLN3nLTRC register, the header reception and the response reception are executed in this order.)
  The LIN slave self-test mode (reception) is executed. In this mode, interrupt are generated, and status and error status are also updated.

- When the reception is completed, the reversed value of the looped-back frame data is stored in the RLN3nLIDB, RLN3nLDBRb (b = 1 to 8), and RLN3nLCBR registers (the data is reversed before being stored because the set value should be compared with the looped-back value). Then, the FTS bit in the RLN3nLTRC register is cleared.
- If the reception fails to complete due to an error, the applicable error flag is set and the FTS bit in the RLN3nLTRC register is cleared.

**Note:** x: Don't care

**Note 1.** The following register settings are not reflected to the operation of the LIN self-test mode.
The LPRS bit in the RLN3nLWBR register, the RLN3nLBRP0 register, the RLN3nLBRP1 register, and the IBS bit in the RLN3nLSC register.
Therefore, those settings are not necessary.

**Note 2.** If necessary, set the related registers described in **Section 6, Interrupt Controller (INTC)**.

**Note 3.** According to the setting of this register, 9.5-Tbit or 10.5-Tbit width break is output from the internal RLIN3nTX.

**Note 4.** When the successful header reception interrupt and the successful response reception interrupt are used in the same interrupt processing, if the software processing of the successful header reception interrupt is not completed before the generation of the successful response reception interrupt, the SHIE bit in the RLN3nLIE register should not be set to 1 (successful header reception interrupt enabled).
The time required from the set of the successful header reception flag to the set of the successful response/wake-up reception flag is calculated by the following formula.

10 × (number of data bytes + 1) [Tbit]

1 Tbit = frequency of 1/LIN communication clock source × 16

#### 18.5.5.6 Terminating LIN Self-Test Mode

To terminate LIN self-test mode, perform the procedure below:

- Write 0 (LIN reset mode) to the OM0 bit in the RLN3nLCUC register.
If the OMM1 and OMM0 bits in the RLN3nLMST register are not $11_B$, write $11_B$ to the OM1 and OM0 bits in the RLN3nLCUC register. After confirming that the OMM1 and OMM0 bits in the RLN3nLMST register have turned $11_B$, change to LIN reset mode.
- Verify the cancelation of LIN self-test mode.
Read the LSTM bit in the RLN3nLSTC register; confirm that it is not 0 (not in LIN self-test)
- Verify the transition to LIN reset mode.
Read the OMM0 bit in the RLN3nLMST register; verify that it is 0 (LIN reset mode).

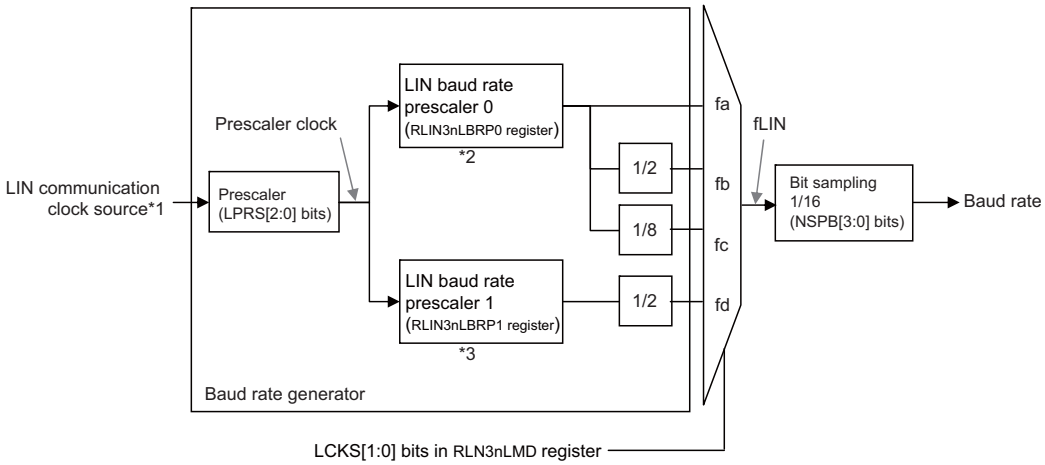
## 18.5.6 Baud Rate Generator

The prescaler clock is obtained by frequency-dividing the LIN communication clock source by the prescaler, and the LIN system clock (fLIN) is obtained by frequency-dividing the prescaler clock by the baud rate generator. The clock obtained by frequency-dividing the LIN system clock (fLIN) by the number of samples is the baud rate. The reciprocal of this baud rate is called the bit time (Tbit).

The LIN/UART interface has two kinds of baud rate generators. The baud rate generators switch over according to the mode used.

### 18.5.6.1 LIN Master Mode

**Figure 18.35** shows a block diagram of baud rate generation in LIN master mode.




Note 1. For the LIN communication clock source, see **Section 12, Clock Controller**.
Note 2. When the value in RLN3nLBRP0 register is N (N = 0 to 255), the clock frequency is divided by N + 1.
Note 3. When the value in RLN3nLBRP1 register is M (M = 0 to 255), the clock frequency is divided by N + 1.

**Figure 18.35 Block Diagram of Baud Rate Generation in LIN Master Mode**

By setting the RLN3nLBRP0 register so that fa is 307200 Hz (= 19200 × 16), the resulting bit rates are fa = 19200 × 16, fb = 9600 × 16, and fc = 2400 × 16. These bit rates are frequency-divided by 16 in the bit timing generator, enabling bit rates of 19200 bps, 9600 bps and 2400 bps, to be generated. Also, by setting the RLN3nLBRP1 register so that fd is 166672 Hz (= 10417 × 16), the resulting bit rate is fd = 10417 × 16. This bit rate is frequency-divided by 16 in the bit timing generator, enabling 10417 bps to be generated.

**Table 18.92** shows examples of baud rate (19200, 9600, 2400, and 10417 bps) generation for each LIN communication clock source frequency, and also the corresponding errors.

Table 18.92 Examples of Baud Rate (19200, 10417, 9600, and 2400 bps) Generation in LIN Master Mode

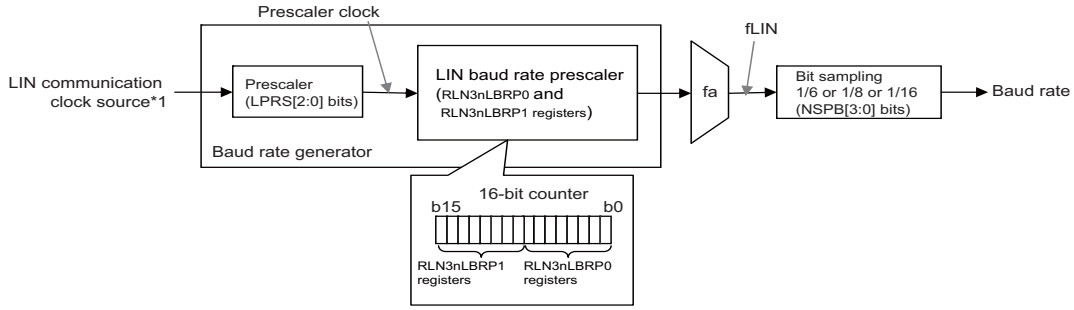
| LIN communication clock source | Prescaler | Baud rate generator 0 (N + 1) frequency-divided | Baud rate generator 1 (M + 1) frequency-divided | System Clock | Baud rate | Error |
|---|---|---|---|---|---|---|
| 40 MHz | 1/1 | 130 | — | fa | 19230.77 | +0.16% |
| | | — | 120 | fd | 10416.67 | –0.003% |
| | | 130 | — | fb | 9615.38 | +0.16% |
| | | 130 | — | fc | 2403.85 | +0.16% |

NOTE

Bit sampling count is 16 sampling (RLN3nLWBR.NSPB[3:0] = $0000_B$).

### 18.5.6.2 LIN Slave Mode

**Figure 18.36** shows a block diagram of baud rate generation in LIN slave mode.




Note 1. For the LIN communication clock source, see **Section 12, Clock Controller**.

Figure 18.36 Block Diagram of Baud Rate Generation in LIN Slave Mode

**Table 18.93** shows examples of baud rate (19200, 10417, 9600, and 2400 bps) generation for each peripheral function clock frequency, and also the corresponding errors.

Table 18.93 Examples of Baud Rate Generation (19200 bps, 10417 bps, 9600 bps, and 2400 bps) in LIN Slave Mode [Fixed Baud Rate]

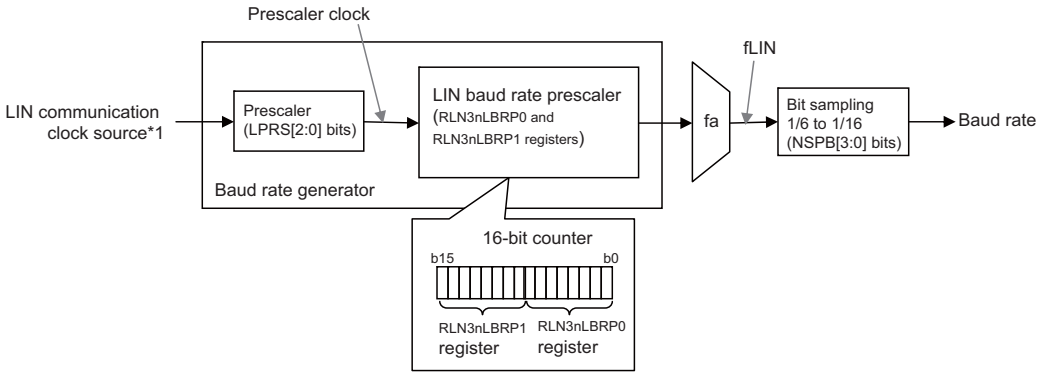
| LIN communication clock | Prescaler | Baud rate generator 01 (MN+1) frequency-divided | Baud rate | Error |
|---|---|---|---|---|
| 40MHz | 1/1 | 130 | 19230.77 | +0.16% |
| | | 240 | 10416.67 | –0.003% |
| | | 260 | 9615.38 | +0.16% |
| | | 1040 | 2403.85 | +0.16% |

NOTE

The bit sampling count in **Table 18.93** is 16 sampling (RLN3nLWBR.NSPB[3:0] = $0000_B$).

### 18.5.6.3 UART Mode

**Figure 18.37** shows a block diagram of baud rate generation in UART mode.




Note 1. For the LIN communication clock source, see **Section 12, Clock Controller**.

**Figure 18.37 Block Diagram of Baud Rate Generation in UART Mode**

UART baud rate is calculated with the following formula:

UART baud rate
= {LIN communication clock source frequency} × (RLN3nLWBR.LPRS[2:0] select clock) ÷ (RLN3nLBRP01 + 1) ÷ {RLN3nLWBR.NSPB[3:0] select count} [bps]

**Table 18.94** lists the examples of baud rate (6600000, 38400, 31250, 19200, 9600, 4800, 2400 and 1200 bps) generation for each LIN communication clock source frequency, and also the corresponding errors.

**Table 18.94 UART Baud Rate Setting Examples (when LIN communication clock source = 40 MHz)**

| UART Baud Rate (Target Baud Rate) | Prescaler | Baud rate generator 01 (MN + 1) frequency-divided | Baud rate | Error |
|---|---|---|---|---|
| 1200 bps | 1/2 | 1042 | 1199.62 | –0.03% |
| 2400 bps | 1/2 | 512 | 2399.23 | –0.03% |
| 4800 bps | 1/2 | 260 | 4807.69 | +0.16% |
| 9600 bps | 1/2 | 130 | 9615.38 | +0.16% |
| 19200 bps | 1/2 | 65 | 19230.77 | +0.16% |
| 31250 bps | 1/2 | 40 | 31250.00 | 0.00% |
| 38400 bps | 1/2 | 33 | 37878.79 | –1.36% |
| 6600000 bps | 1/1 | 1 | 6666666.67 | –1.36% |

**NOTE**

The bit sampling count for each baud rate is written below.

6.6Mbps: 6 sampling (RLN3nLWBR.NSPB[3:0] = $0101_B$)

other than 6.6Mbps: 16 sampling (RLN3nLWBR.NSPB[3:0] = $1111_B$)

### 18.5.7 Noise Filter

The LIN/UART interface has a noise filter for reducing erroneous receiving of data due to noise. By setting the LRDNFS bit in the RLN3nLMD register to 0 (to use the noise filter), the noise filter is activated. The noise filter samples the level of the synchronized RLIN3nRX with the prescaler clock, and outputs the sampling value determined by a 3-sampling majority rule. The value of each bit of the receive data is determined based on the noise filter output.

**Figure 18.38** shows the configuration of the noise filter, **Figure 18.39** shows an example of a noise filter circuit, and **Figure 18.40** shows the determination of the received data when the noise filter is used.

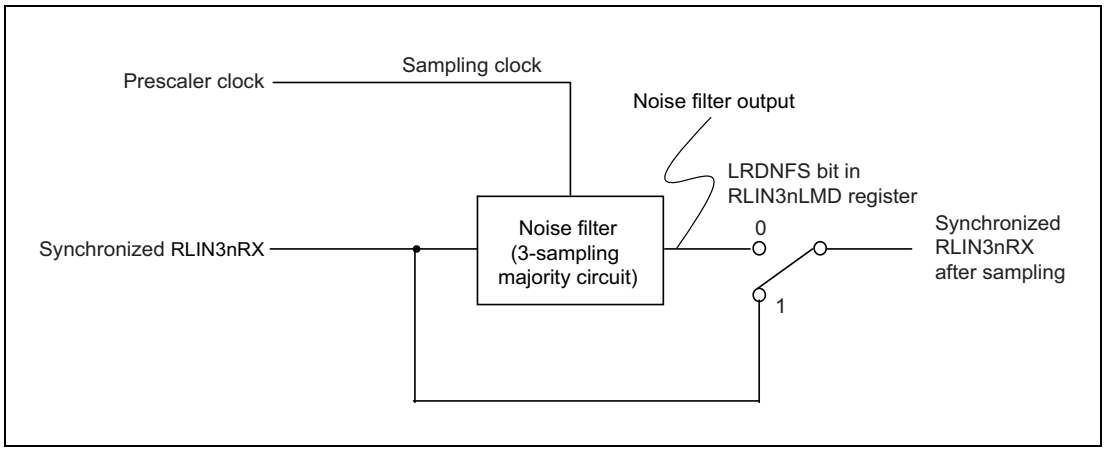



**Figure 18.38 Configuration of Noise Filter**

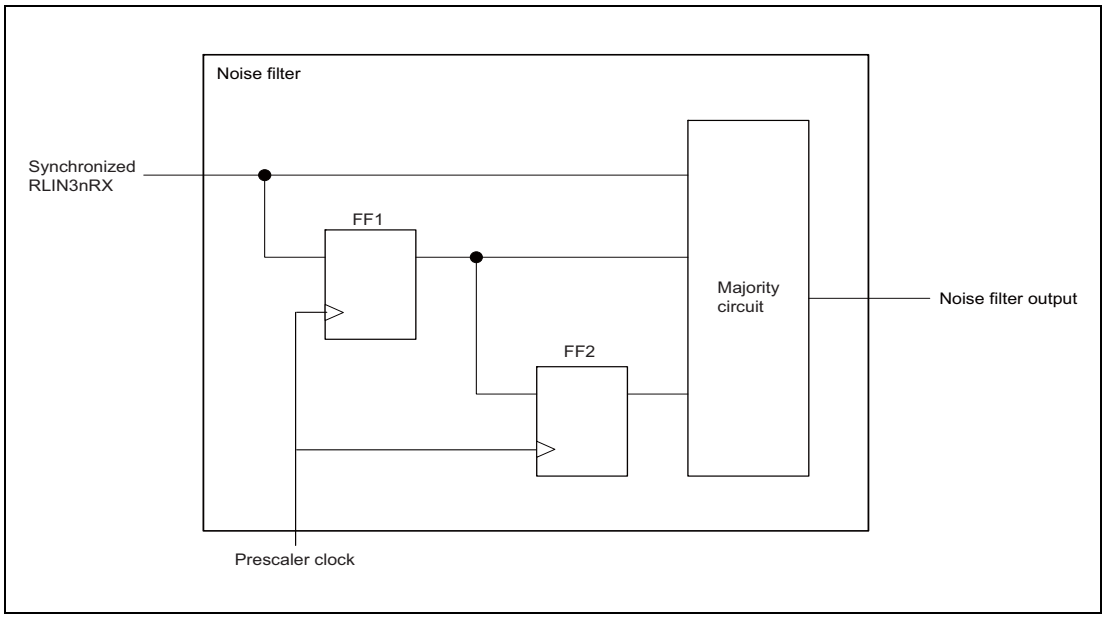



**Figure 18.39 Example of Noise Filter Circuit**

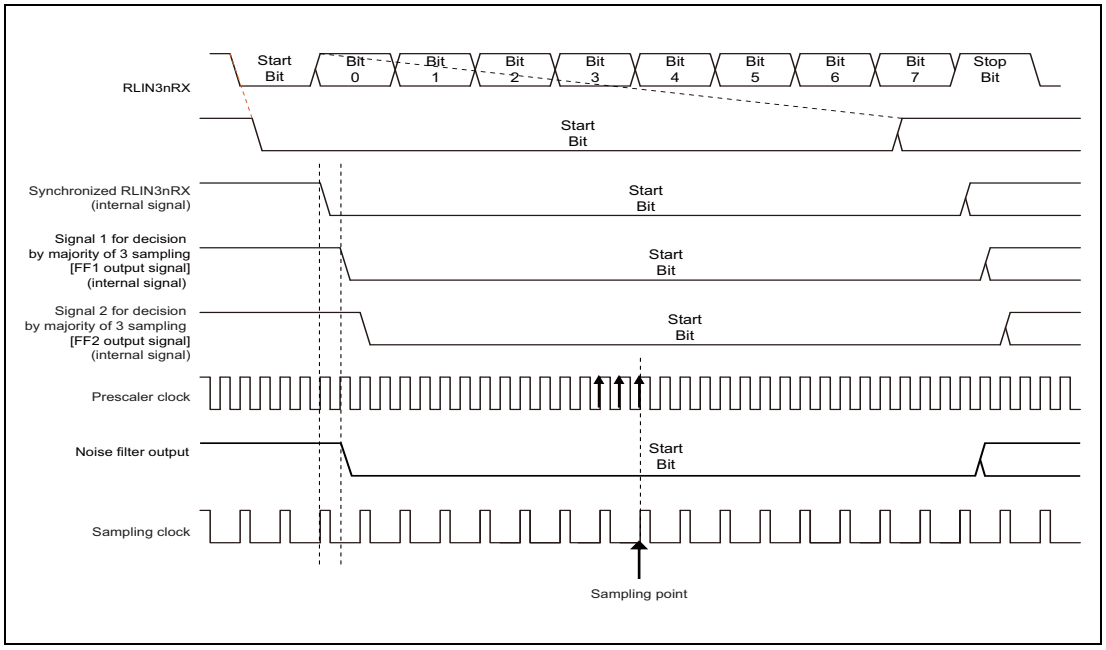



**Figure 18.40 Determination of Received Data when Noise Filter is Used**

## 18.5.8 Limited Reset and Module stand-by

The RLIN3n can be reset by limited reset from SYSCTRL. The Limited Reset is functionally equivalent to a hardware reset. Software must ensure that RLIN3n is halted (LIN reset mode or idle mode). (See **Section 18.5.2, LIN Reset Mode** or **Section 18.5.3, LIN Mode)**

The RLIN3n clock can be disabled by the SYSCTRL module stand-by function. Software must ensure that RLIN3n is LIN reset mode or idle mode if module stand-by enable.

## 18.6 Difference among P1M-C, P1H-C and P1H-CE

Table 18.95 Different specification of RLIN3n

| Device | P1M-C and P1H-C (4MB, BGA-156) | P1H-C (4MB, BGA-292) and P1H-C (8MB) | P1H-CE |
|---|---|---|---|
| RLIN3n channel | RLIN3n: 2 | RLIN3n: 4 | RLIN3n: 4 |

# Section 19 CAN Controller (MCAN)

The Features described in this chapter are only valid for the following devices:

- R7F701370B
- R7F701371
- R7F701372A
- R7F701396A
- R7F701373A
- R7F701397A
- R7F701374A

For the MCAN description of other devices, please refer to **Section 37, CAN Controller (MCAN)**
According to Bosch CAN-FD Specification V1.0

ISO Draft - ISO 11898-1:2015

## 19.1 Features

The CAN controller (M(TT)CAN) is used to communicate via the CAN bus, as define in ISO11898-1:2015. It can also support CAN FD (CAN with Flexible Data-rate). It covers the functionality of the Data Link Layer (DLL) and Medium Access Control (MAC). To enable the hardware to communicate, the physical layer must be provided externally, i.e., by connecting a CAN transceiver.

**Table 19.1 M_CAN Specifications (1/2)**

| Item | Specification |
|---|---|
| Communication | CAN functionality conform to ISO 11898-1:2015 |
| Data transfer rate | Up to 1 Mbps, individually for each CAN channel for CAN FD, up to 8 Mbps |
| CAN channels | MTTCAN0, MCAN0, MCAN1, MCAN2 |
| Selectable ID type | 11-bit Standard ID |
| | 11-bit Standard ID + 18-bit Extended ID |
| Message Buffer | Up to 64 dedicated Receive Buffers |
| | Up to 32 dedicated Transmit Buffers |
| FIFO number | Two configurable Receive FIFOs<br>Configurable Transmit FIFO<br>Configurable Transmit Queue<br>Configurable Transmit Event FIFO |
| RX System | Scalable RX FIFO structures, with up to 64 CAN Buffers per FIFO<br>RX timestamp<br>RX FIFO Timeout Interrupt<br>FIFO filling level Interrupt |
| TX System | FIFO filling level supervision (interrupt)<br>Support for transmit cancellation to avoid<br>“Inner priority inversion”<br>Combined Message Buffer & TX FIFO and TX Queue Concept<br>Dedicated TX message buffers for high-priority messages<br>ID prioritization between TX buffers, TX Queue buffers and oldest TX FIFO element<br>Transmit pause to separate two consecutive TX messages |

Table 19.1 M_CAN Specifications (2/2)

| Item | Specification |
|---|---|
| Enhanced reception filtering | Support of 11bit and 29bit CAN identifier, each filter element is configurable for acceptance/ rejection |
| | Programmable 29 bit CAN identifier acceptance filter mask for each entry |
| | Each acceptance filter element targets FIFO 0 or 1 or a dedicated RX Buffer |
| | Every FIFO or RX Buffer filter element can be used as a from-to range filter, as a filter for one or two dedicated IDs or as a classic bit mask filter |
| | Each filter element can be enabled/disabled individually |
| TT-CAN support | MTTCAN0 supports TT-CAN level 2 according to ISO11898-4 |
| CAN-FD support | Variable Data Phase Speed up to 8 Mbit/s, variable Data Buffer size up to 64 Bytes/Frame |
| Timer | Time Stamp function |
| Power down function | Local Power Down modes |
| AUTOSAR requirements | Supports all AUTOSAR requirements<br>Like Transmit Abort Interrupt, non-waiting processing<br>Functionality and more than 2 TX Buffers with prioritization<br>Supports *Pretended Networking* of AUTOSAR |
| Self-testing | External and internal loop back |

ISO CAN FD 11898-1:2015

### 19.1.1 Number of Units and Channels

Table 19.2 Number of MTTCAN/MCAN channel

| | Device | | | | |
|---|---|---|---|---|---|
| Macro | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE | Description |
| MTTCAN0 | 1*[1] | 1*[1] | 1 | 1 | MTTCAN0 supports TT-CAN level 2 according to ISO11898-4 |
| MCAN0 | 1 | 1 | 1 | 1 | — |
| MCAN1 | 1 | 1 | 1 | 1 | — |
| MCAN2 | 0 | 0 | 1 | 1 | — |

Note 1. P1M-C, P1H-C (4MB, BGA-156) do not support MTTCAN0EVT, MTTCAN0RTP, MTTCAN0SOC, MTTCAN0SWT, and MTTCAN0TMP.

### 19.1.2 Register Base Address

MTTCAN/MCAN base addresses are listed in the following table. MTTCAN/MCAN register addresses are given as offsets from the base addresses in general.

Table 19.3 Register Base Address

| Base Address Name | Base Address |
|---|---|
| <MTTCAN0_base> | FFD3 $0000_H$ |
| <MCAN0_base> | FFEF $0000_H$ |
| <MCAN1_base> | FFD3 $1000_H$ |
| <MCAN2_base> | FFEF $1000_H$ |

### 19.1.3 Clock Supply

Clock supply by and to MTTCAN/MCAN is listed in the following table.

**Table 19.4 Clock Supply**

| Unit Name | Clock for the Unit | Supply Clock Name |
|---|---|---|
| MTTCAN0/ MCANn | H-Bus interface clock*1 (Host clock) | CLK_HSB |
| | CAN protocol layer clock*2 (CAN clock / m_can_cclk/ m_ttcan_cclk) | CLKP_H2 |

**Note:** The following condition must be fulfilled: CLK_HSB >= CLKP_H2

Note 1. H-bus interface clock:
The minimum frequency is **Table 19.5, Minimum MCAN / MTTCAN clock speed**.

Note 2. CAN protocol layer clock
- For maximum data rate, CLKP_H2 must be n x 1 MHz with n >= 8.
- For CAN-FD operation with bit rates above 1Mbit/s CLKP_H2 must be 2^n x 20MHz, with n = 0, 1, 2.
- For low power mode with limited throughput it can be reduce to CLK_HSB/3. This limits the throughput to 333Kbps.

**Table 19.5 Minimum MCAN / MTTCAN clock speed**

| | P1M-C | P1H-C (4MB) | P1H-C (8MB) |
|---|---|---|---|
| Number of channels | 3 | 3 | 4 |
| Non CANFD | 17.5 MHz | | |
| CANFD | | | |

ISO CANFD ISO 11898-1:2015

### 19.1.4 RAM areas for MCAN

**Table 19.6 RAM areas for MCAN**

| RAM Name | RAM Area |
|---|---|
| MTTCAN RAM area | FFD3 $8000_H$ to FFD3 $9FFF_H$ |
| MCAN0 RAM area | FFEF $8000_H$ to FFEF $9FFF_H$ |
| MCAN1 RAM area | FFD3 $A000_H$ to FFD3 $BFFF_H$ |
| MCAN2 RAM area | FFEF $A000_H$ to FFEF $BFFF_H$ |

### 19.1.5 Interrupt Requests

MTTCAN/MCAN interrupt requests are listed in the following table.

**Table 19.7 Interrupt list**

| Unit Name | Interrupt Name | Interrupt Number | DMA Number | Description |
|---|---|---|---|---|
| MTTCAN0 | INTMTTCANI0 | 172 | — | MTTCAN0 interrupt 0 |
| MTTCAN0 | INTMTTCANI1 | 173 | — | MTTCAN0 interrupt 1 |
| MTTCAN0 | INTMTTCANFE | 174 | — | MTTCAN0 filter event 1 |
| MCAN0 | INTMCAN0I0 | 175 | — | MCAN0 interrupt 0 |
| MCAN0 | INTMCAN0I1 | 176 | — | MCAN0 interrupt 1 |
| MCAN0 | INTMCAN0FE | 177 | — | MCAN0 filter event 1 |
| MCAN1 | INTMCAN1I0 | 178 | — | MCAN1 interrupt 0 |
| MCAN1 | INTMCAN1I1 | 179 | — | MCAN1 interrupt 1 |
| MCAN1 | INTMCAN1FE | 180 | — | MCAN1 filter event 1 |
| MCAN2 | INTMCAN2I0 | 181 | — | MCAN2 interrupt 0 |
| MCAN2 | INTMCAN2I1 | 182 | — | MCAN2 interrupt 1 |
| MCAN2 | INTMCAN2FE | 183 | — | MCAN2 filter event 1 |

**Table 19.8 Internal Error Signal**

| Unit Name | Interrupt for Unit | Description | Connected to |
|---|---|---|---|
| MTTCAN0 | mttcan0_ecc_err | MTTCAN0 error signal | ECM |
| MCAN0 | mcan0_ecc_err | MCAN0 error signal | ECM |
| MCAN1 | mcan1_ecc_err | MCAN1 error signal | ECM |
| MCAN2 | mcan2_ecc_err | MCAN2 error signal | ECM |

ISO CANFD ISO 11898-1:2015

### 19.1.6 External Input / Output Pins

MTTCAN/MCAN has following external pins for each channel.

**Table 19.9 External Input/Output Pins**

| Channel | I/O | Pin name of RH850/P1x-C | Function |
|---|---|---|---|
| MTTCAN0 | I | MTTCAN0RX | MTTCAN0 receive data input |
| | O | MTTCAN0TX | MTTCAN0 transmit data output |
| | I | MTTCAN0SWT | MTTCAN0 stop watch trigger |
| | I | MTTCAN0EVT | MTTCAN0 event trigger |
| | O | MTTCAN0RTP | MTTCAN0 register time mark interrupt pulse |
| | O | MTTCAN0TMP | MTTCAN0 trigger time mark interrupt pulse |
| | O | MTTCAN0SOC | MTTCAN0 start of cycle |
| MCAN0 | I | MCAN0RX | MCAN0 receive data input |
| | O | MCAN0TX | MCAN0 transmit data output |
| MCAN1 | I | MCAN1RX | MCAN1 receive data input |
| | O | MCAN1TX | MCAN1 transmit data output |
| MCAN2 | I | MCAN2RX | MCAN2 receive data input |
| | O | MCAN2TX | MCAN2 transmit data output |
| MCKDV | O | BHPDGRCLK0 | Degrading clock output for MCAN |

## 19.2 Overview

### 19.2.1 Functional overview

M(TT)CAN has following features

- M(TT)CAN functionality conform to ISO 11898-1:2015
- M(TT)CAN supports local Power Down modes
- M(TT)CAN supports all AUTOSAR requirements
  - Transmit Abort Interrupt
  - Non-waiting processing functionality
  - Include more than 2 TX Buffers prioritization
- M(TT)CAN supports several measures for self-testing:external and internal loop back
  - For this reason, the port structure does not have additional functionality for self-testing
  - The usage of loop back functionality for safety requirements is described in **Section 28, Functional Safety**
- M(TT)CAN contains an improved RX System
  - Scalable RX FIFO structures, with up to 64 CAN Buffers per FIFO
  - RX timestamp
  - RX FIFO Timeout Interrupt
  - FIFO filling level Interrupt
- M(TT)CAN contains an improved TX System
  - Variable amount of 0 to 32 “classical” TX Buffers
  - Additional, size configurable TX FIFO
  - Additional, size configurable TX Queue
  - FIFO filling level supervision (interrupt)
  - Support for transmit cancellation to avoid “inner priority inversion”
  - Configurable TX Event history: For each transmitted message both ID and timestamp are written into a history
  - TX-Event FIFO (readable by CPU using polling or interrupts) containing 0 to 32 events
  - Combined Message Buffer & TX FIFO and TX Queue Concept
  - Dedicated TX message buffers for high-priority messages
  - ID prioritization between TX buffers, TX Queue buffers and oldest TX FIFO element
  - Transmit pause to separate two consecutive TX messages
- M(TT)CAN contains an enhanced reception filtering
  - Support of 11bit and 29bit CAN identifier, each filter element is configurable for acceptance/ rejection
  - Each acceptance filter element targets FIFO 0 or 1 or a dedicated RX Buffer
  - Every FIFO or RX Buffer filter element can be used as a from-to range filter, as a filter for one or two dedicated
  - IDs or as a classic bit mask filter

ISO CANFD ISO 11898-1:2015

  - Each filter element can be enabled/disabled individually
  - Programmable 29 bit CAN identifier acceptance filter mask for each entry
- MTTCAN supports TT_CAN level 2 according to ISO11898-4
- MTTCAN supports additional synchronization input signals
  - MTTCAN supports additional status output signals
- Data transfer rate is up to 1Mbps, individually for each CAN channel. For CAN FD, up to 8Mbps
- M(TT)CAN supports Pretended Networking of AUTOSAR
  - This can be achieved by using the dedicated additional interrupt assigned to messages of filter group 1
- Selectable ID type
  - 11-bit Standard ID
  - 11-bit Standard ID + 18-bit Extended ID
- M(TT)CAN are supported by the debug system by means of a debug stop signal, which suppresses the register content modification by reading of accesses of the debugger
- M(TT)CAN supports Time Stamp function
- M(TT)CAN are supporting CAN-FD with 64 data bytes and flexible data rate

ISO CANFD ISO 11898-1:2015

### 19.2.2 Block Diagram

Following block diagram shows MTTCAN/MCAN block diagram. P1H-C (8MB) and P1H-CE support MCAN2. P1M-C and P1H-C (4MB) does not support MCAN2.

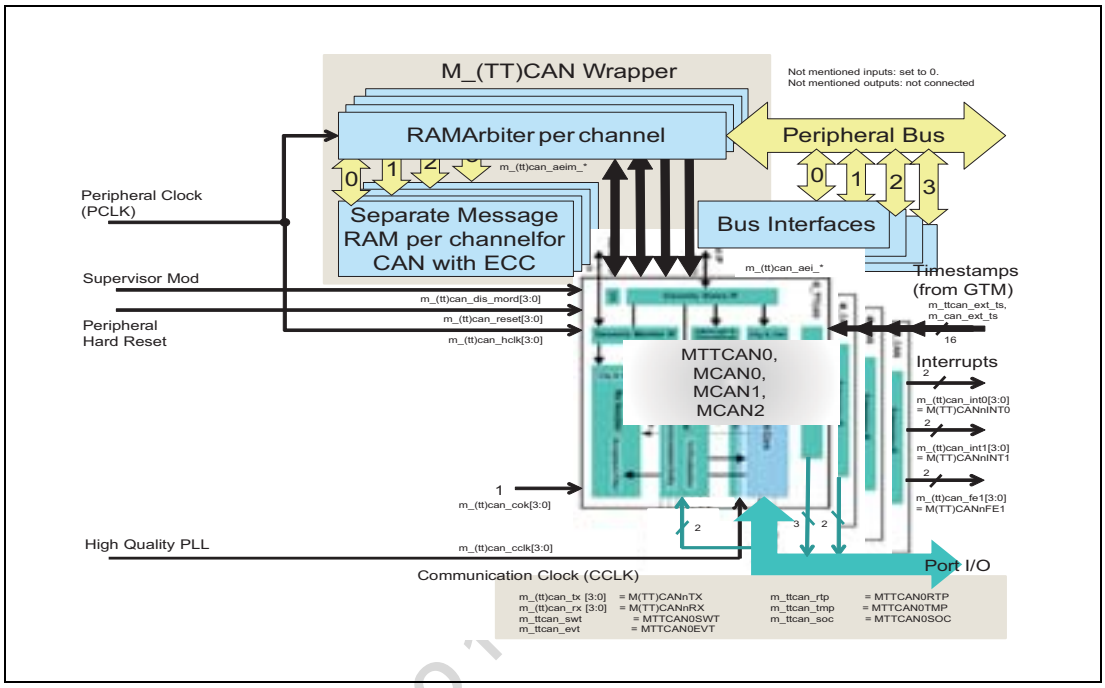



Figure 19.1 MCAN/MTTCAN block diagram

- M_TTCAN (MTTCAN0)
  8K RAM for messages, events and filters; no CCU implemented
  additional input signals MTTCAN0SWT, MTTCAN0EVT;*1
  additional output signals MTTCAN0RTP, MTTCAN0TMP, MTTCAN0SOC;*1
  timestamps from GTM IP
- M_CAN (MCAN0)
  8K RAM for messages, events and filters; common clocking with MTTCAN0
  timestamps from GTM IP
- M_CAN (MCAN1)
  8K RAM for messages, events and filters; common clocking with MTTCAN0
  timestamps from GTM IP
- M_CAN (MCAN2)
  8K RAM for messages, events and filters; common clocking with MTTCAN0
  timestamps from GTM IP
  P1M-C and P1H-C (4MB) don't support MCAN2.

  *m_(tt) can_cok fixed "1"*2

**Note 1.** These signals are not supported in P1M-C, P1H-C (4MB, BGA-156).

**Note 2.** This signal of M_(TT) CAN indicates that a proper clock supply is available. In devices P1M-C and P1H-C (incl. P1H-CE), clock is provided permanently so that M_(TT) CAN is enabled for transmission by this signal without precondition.

# 19.3 Registers

## 19.3.1 List of Registers

MTTCAN/MCAN registers are listed in the following table.

**Table 19.10 Register list**

| Address | Register name | Description | Access Size[bit] | Initial Value | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | Other |
| *1 | *1 | MTTCAN0 | 32 | *1 | PBG3#1.PG3-MCANT | — |
| *1 | *1 | MCAN0 | 32 | *1 | PBG1#1.PG1-MCAN0 | — |
| *1 | *1 | MCAN1 | 32 | *1 | PBG3#1.PG3-MCAN1 | — |
| *1 | *1 | MCAN2 | 32 | *1 | PBG1#1.PG1-MCAN2 | — |

Note 1. For details of MTTCAN/MCAN-IP registers, please see **Table 19.71, M_TTCAN Register Map** and **Table 19.12, M_CAN Register Map**.

**Table 19.11 Register Reset Condition**

| Register Name | Reset condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset1 | System Reset2 | Application Reset1 | Limited Reset |
| All registers | √ | √ | √ | √ | √ |

ISO CANFD ISO 11898-1:2015

# 19.4 Operation

## 19.4.1 Procedure of Module Standby and Limited Reset

This module supports module standby and limited reset functions. Before these functions are enabled, all of the followings must be ensured:

- Module standby
  The M(TT)CAN can be set into power down mode controlled by CC Control Register M(TT)CANnCCCR.CSR. As long as the clock stop request is active, bit M(TT)CANnCCCR.CSR is read as one. When all pending transmission requests have completed, the M(TT)CAN waits until bus idle state is detected. Then the M(TT)CAN sets M(TT)CANnCCCR.INIT to one to prevent any further CAN transfers. Now the M(TT)CAN acknowledges that it is ready for power down by setting M(TT)CANnCCCR.CSA to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to M(TT)CANnCCCR.INIT will have no effect. Now the module clock may be switched off. To leave power down mode, the application has to turn on the module clocks before resetting the CC Control Register flag M(TT)CANnCCCR.CSR. The M_CAN will acknowledge this by resetting M(TT)CANnCCCR.CSA. Afterwards, the application can restart CAN communication by resetting bit M(TT)CANnCCCR.INIT. Refer to the following flow chart.
- Limited Reset
  The M(TT)CAN can be reset by limited reset from SYSCTRL. The Limited Reset is functionally equivalent to a hardware reset. After limited reset, the registers of the M(TT)CAN hold the reset values. Additionally the Bus_Off state is reset and the output “m_(tt)can_tx” is set to recessive (HIGH). The value $0001_H$ (M(TT)CANnCCCR.INIT = ‘1’) in the CC Control Register enables software initialization. The M(TT)CAN does not influence the CAN bus until the CPU resets M(TT)CANnCCCR.INIT to ‘0’. Refer to the following flow chart.

ISO CANFD ISO 11898-1:2015

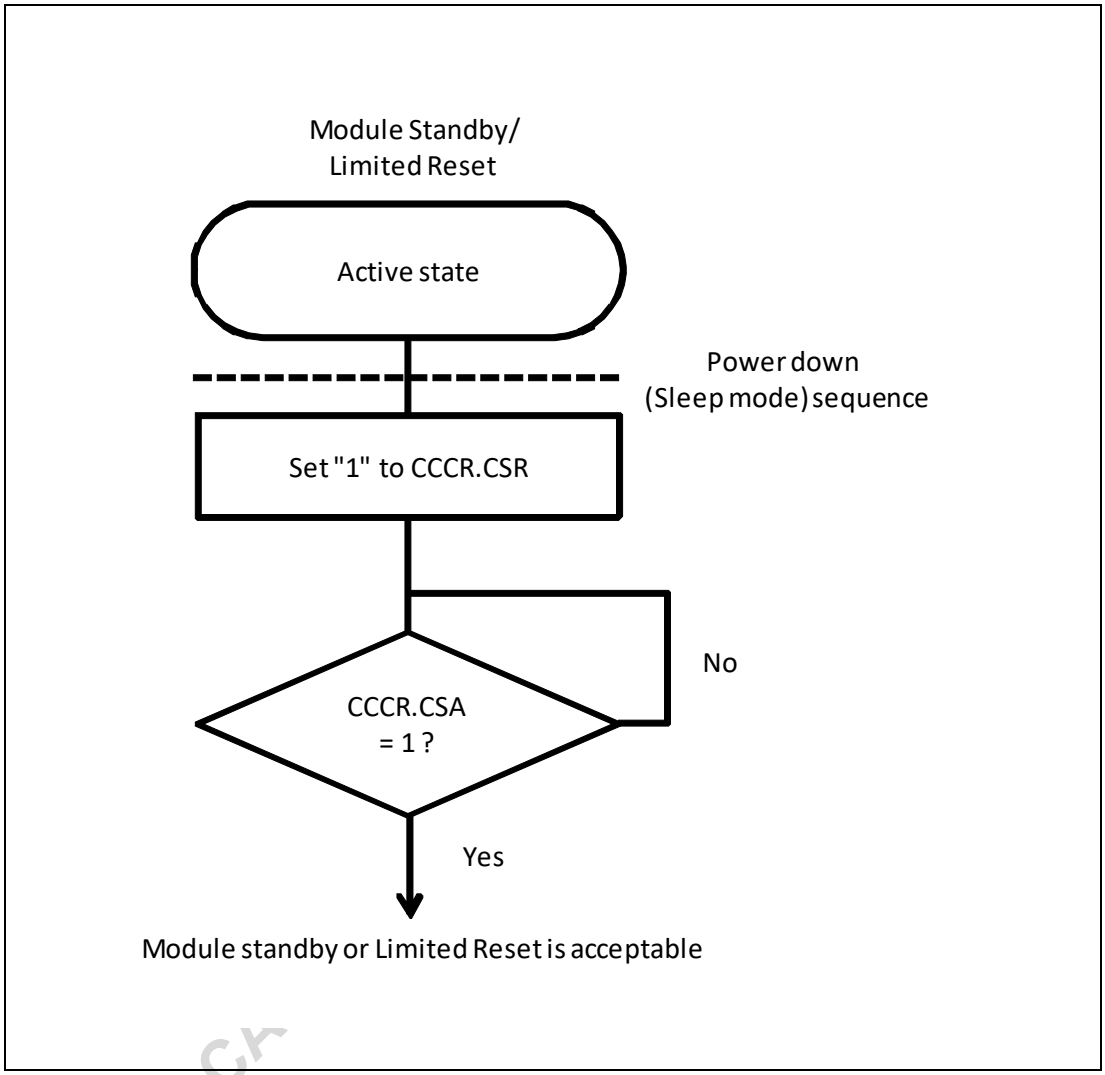



**Figure 19.2 Module Standby/Limited Reset procedure**

# 19.5 MCAN

## 19.5.1 Overview

The M_CAN performs communication according to ISO11898-1:2015. Additional transceiver hardware is required for connection to the physical layer.

The message storage is intended to be a single-ported Message RAM outside of the module. It is connected to the M_CAN via the Generic Master Interface.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN Core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN Core as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

ISO CANFD ISO 11898-1:2015

### 19.5.1.1 Features

- Conform with ISO 11898-1:2015
- CAN FD with up to 64 data bytes supported
- CAN Error Logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signalling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains (CAN clock and Host clock)
- Power-down support

### 19.5.1.2 Block Diagram

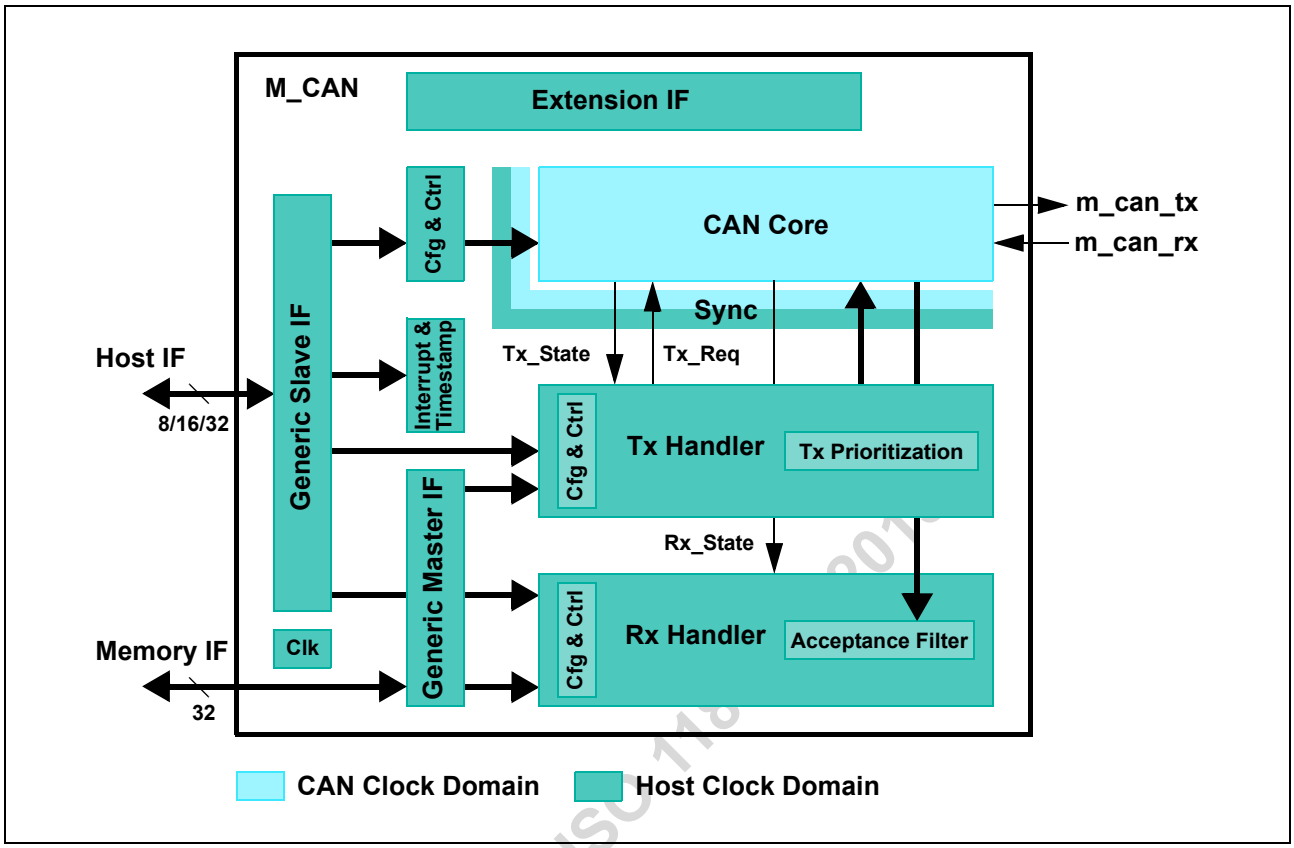



Figure 19.3 M_TTCAN Block Diagram

**CAN Core:**

CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.

**Sync:**

Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.

**Clk:**

Synchronizes reset signal to the Host clock domain and to the CAN clock domain.

**Cfg & Ctrl:**

CAN Core related configuration and control bits.

**Interrupt & Timestamp:**

Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation. An externally generated 16-bit vector may substitute the integrated 16-bit CAN bit time counter for receive and transmit timestamp generation.

ISO CANFD ISO 11

**Tx Handler:**

Controls the message transfer from the external Message RAM to the CAN Core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.

**Rx Handler:**

Controls the transfer of received messages from the CAN Core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

**Generic Slave Interface:**

Connects the M_CAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8/16/32-bit bus to support a wide range of interconnection structures.

**Generic Master Interface:**

Connects the M_CAN to a local 32-bit Message RAM. The implemented Message RAM size is 2K • 32 bit.

Extension Interface:

All flags from the Interrupt Register MCANnIR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M_CAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

ISO CANFD ISO 11898-1:2015

### 19.5.1.3 Dual Clock Sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain m_can_hclk (CLK_HSB). Due to the high precision clocking requirements of the CAN Core, a separate clock without any modulation has to be provided as m_can_cclk (CLKP_H2).

Within the M_TTCAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

**NOTE**

In order to achieve a stable function of the M_TTCAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

### 19.5.1.4 Dual Interrupt Lines

The module provides two interrupt lines. Interrupts can be routed either to m_can_int0 (INTMCANnI0) or to m_can_int1 (INTMCANnI1). By default all interrupts are routed to interrupt line m_can_int0 (INTMCANnI0). By programming MCANnILE.EINT0 and MCANnILE.EINT1 the interrupt lines can be enabled or disabled separately.

## 19.5.2 Programmer's Model

### 19.5.2.1 Hardware Reset Description

After hardware reset, the registers of the M_TTCAN hold the reset values listed in **Table 19.12**. Additionally the Bus_Off state is reset and the output m_can_tx is set to recessive (HIGH). The value $0001_H$ (MCANnCCCR.INIT = '1') in the CC Control Register enables software initialization. The M_TTCAN does not influence the CAN bus until the CPU resets MCANnCCCR.INIT to '0'.

### 19.5.2.2 Register Map

The M_TTCAN module allocates an address space of 256 bytes. All registers are organized as 32-bit registers. The M_CAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8 bit (byte access), 16 bit (half-word access), or 32 bit (word access). Write access by the Host CPU to registers/bits marked with "P = Protected Write" is possible only with MCANnCCCR.CCE = '1' AND MCANnCCCR.INIT = '1'. There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

**Table 19.12 M_CAN Register Map (1/2)**

| ADDRESS | SYMBOL | NAME | page | RESET | ACC |
|---|---|---|---|---|---|
| <MCANn_base> + $000_H$ | MCANnCREL | Core Release Register | 1000 | 3215 0320 | R |
| <MCANn_base> + $004_H$ | MCANnENDN | Endian Register | 1001 | 8765 4321 | R |
| <MCANn_base> + $00C_H$ | MCANnDBTP | Data Bit Timing & Prescaler Register | 1002 | 0000 0A33 | RP |
| <MCANn_base> + $010_H$ | MCANnTEST | Test Register | 1004 | 0000 0000 | RP |
| <MCANn_base> + $014_H$ | MCANnRWD | RAM Watchdog | 1005 | 0000 0000 | RP |
| <MCANn_base> + $018_H$ | MCANnCCCR | CC Control Register | 1006 | 0000 0001 | RWPp |
| <MCANn_base> + $01C_H$ | MCANnNBTP | Nominal Bit Timing & Prescaler Register | 1008 | 0600 0A03 | RP |
| <MCANn_base> + $020_H$ | MCANnTSCC | Timestamp Counter Configuration | 1009 | 0000 0000 | RP |
| <MCANn_base> + $024_H$ | MCANnTSCV | Timestamp Counter Value | 1010 | 0000 0000 | RC |
| <MCANn_base> + $028_H$ | MCANnTOCC | Timeout Counter Configuration | 1011 | FFFF 0000 | RP |
| <MCANn_base> + $02C_H$ | MCANnTOCV | Timeout Counter Value | 1012 | 0000 FFFF | RC |
| <MCANn_base> + $030_H$ to $03C_H$ | | reserved (4) | | 0000 0000 | R |
| <MCANn_base> + $040_H$ | MCANnECR | Error Counter Register | 1013 | 0000 0000 | RX |
| <MCANn_base> + $044_H$ | MCANnPSR | Protocol Status Register | 1014 | 0000 0707 | RXS |
| <MCANn_base> + $048_H$ | MCANnTDCR | Transmitter Delay Compensation Register | | 0000 0000 | RP |
| <MCANn_base> + $04C_H$ | | reserved (1) | | 0000 0000 | R |
| <MCANn_base> + $050_H$ | MCANnIR | Interrupt Register | 1017 | 0000 0000 | RW |
| <MCANn_base> + $054_H$ | MCANnIE | Interrupt Enable | 1020 | 0000 0000 | RW |
| <MCANn_base> + $058_H$ | MCANnILS | Interrupt Line Select | 1022 | 0000 0000 | RW |
| <MCANn_base> + $05C_H$ | MCANnILE | Interrupt Line Enable | 1024 | 0000 0000 | RW |
| <MCANn_base> + $060_H$ to $07C_H$ | | reserved (8) | | 0000 0000 | R |
| <MCANn_base> + $080_H$ | MCANnGFC | Global Filter Configuration | 1025 | 0000 0000 | RP |
| <MCANn_base> + $084_H$ | MCANnSIDFC | Standard ID Filter Configuration | 1026 | 0000 0000 | RP |
| <MCANn_base> + $088_H$ | MCANnXIDFC | Extended ID Filter Configuration | 1027 | 0000 0000 | RP |
| <MCANn_base> + $08C_H$ | | reserved (1) | | 0000 0000 | R |
| <MCANn_base> + $090_H$ | MCANnXIDAM | Extended ID AND Mask | 1028 | 1FFF FFFF | RP |
| <MCANn_base> + $094_H$ | MCANnHPMS | High Priority Message Status | 1029 | 0000 0000 | R |
| <MCANn_base> + $098_H$ | MCANnNDAT1 | New Data 1 | 1030 | 0000 0000 | RW |
| <MCANn_base> + $09C_H$ | MCANnNDAT2 | New Data 2 | 1031 | 0000 0000 | RW |
| <MCANn_base> + $0A0_H$ | MCANnRXF0C | Rx FIFO 0 Configuration | 1032 | 0000 0000 | RP |

ISO CANFD ISO 11898-1:2015

Table 19.12 M_CAN Register Map (2/2)

| ADDRESS | SYMBOL | NAME | page | RESET | ACC |
|---|---|---|---|---|---|
| <MCANn_base> + $0A4_H$ | MCANnRXF0S | Rx FIFO 0 Status | 1033 | 0000 0000 | R |
| <MCANn_base> + $0A8_H$ | MCANnRXF0A | Rx FIFO 0 Acknowledge | 1034 | 0000 0000 | RW |
| <MCANn_base> + $0AC_H$ | MCANnRXBC | Rx Buffer Configuration | 1035 | 0000 0000 | RP |
| <MCANn_base> + $0B0_H$ | MCANnRXF1C | Rx FIFO 1 Configuration | 1036 | 0000 0000 | RP |
| <MCANn_base> + $0B4_H$ | MCANnRXF1S | Rx FIFO 1 Status | 1037 | 0000 0000 | R |
| <MCANn_base> + $0B8_H$ | MCANnRXF1A | Rx FIFO 1 Acknowledge | 1038 | 0000 0000 | RW |
| <MCANn_base> + $0BC_H$ | MCANnRXESC | Rx Buffer / FIFO Element Size Configuration | 1039 | 0000 0000 | RP |
| <MCANn_base> + $0C0_H$ | MCANnTXBC | Tx Buffer Configuration | 1040 | 0000 0000 | RP |
| <MCANn_base> + $0C4_H$ | MCANnTXFQS | Tx FIFO/Queue Status | 1041 | 0000 0000 | R |
| <MCANn_base> + $0C8_H$ | MCANnTXESC | Tx Buffer Element Size Configuration | 1042 | 0000 0000 | RP |
| <MCANn_base> + $0CC_H$ | MCANnTXBRP | Tx Buffer Request Pending | 1043 | 0000 0000 | R |
| <MCANn_base> + $0D0_H$ | MCANnTXBAR | Tx Buffer Add Request | 1044 | 0000 0000 | RW |
| <MCANn_base> + $0D4_H$ | MCANnTXBCR | Tx Buffer Cancellation Request | 1045 | 0000 0000 | RW |
| <MCANn_base> + $0D8_H$ | MCANnTXBTO | Tx Buffer Transmission Occurred | 1046 | 0000 0000 | R |
| <MCANn_base> + $0DC_H$ | MCANnTXBCF | Tx Buffer Cancellation Finished | 1047 | 0000 0000 | R |
| <MCANn_base> + $0E0_H$ | MCANnTXBTIE | Tx Buffer Transmission Interrupt Enable | 1047 | 0000 0000 | RW |
| <MCANn_base> + $0E4_H$ | MCANnTXBCIE | Tx Buffer Cancellation Finished Interrupt Enable | 1048 | 0000 0000 | RW |
| <MCANn_base> + $0E8_H$ to $0EC_H$ | | reserved (2) | | 0000 0000 | R |
| <MCANn_base> + $0F0_H$ | MCANnTXEFC | Tx Event FIFO Configuration | 1049 | 0000 0000 | RP |
| <MCANn_base> + $0F4_H$ | MCANnTXEFS | Tx Event FIFO Status | 1050 | 0000 0000 | R |
| <MCANn_base> + $0F8_H$ | MCANnTXEFA | Tx Event FIFO Acknowledge | 1051 | 0000 0000 | RW |
| <MCANn_base> + $0FC_H$ | | reserved (1) | | 0000 0000 | R |

**Note:** R = Read, S = Set on read, X = Reset on read, W = Write, P = Protected write, p = Protected set, C = Clear/preset on write

ISO CANFD ISO 11898-1:2015

### 19.5.2.3 Registers

#### (1) MCANnCREL — Core Release Register

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $000_H$

**Value after reset:** $3215\ 0320_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | REL[3:0] | | | | STEP[3:0] | | | | SUBSTEP[3:0] | | | | YEAR[3:0] | | | |
| Value after reset | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MON[7:0] | | | | | | | | DAY[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

ISO CANFD SC 11898-1:2015

**Table 19.13 MCANnCREL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | REL[3:0] | Core Release<br>One digit, BCD-coded. |
| 27 to 24 | STEP[3:0] | Step of Core Release<br>One digit, BCD-coded. |
| 23 to 20 | SUBSTEP[3:0] | Sub-step of Core Release<br>One digit, BCD-coded. |
| 19 to 16 | YEAR[3:0] | Time Stamp Year<br>One digit, BCD-coded. This field is set by generic parameter on M_CAN synthesis. |
| 15 to 8 | MON[7:0] | Time Stamp Month<br>Two digits, BCD-coded. This field is set by generic parameter on M_CAN synthesis. |
| 7 to 0 | DAY[7:0] | Time Stamp Day<br>Two digits, BCD-coded. This field is set by generic parameter on M_CAN synthesis. |

**Table 19.14 Coding of Revisions**

| Release | Step | SubStep | Year | Month | Day | Name |
|---|---|---|---|---|---|---|
| 3 | 2 | 1 | 5 | 03 | 20 | Revision 3.2.1, Date 2015/03/20 |

### (2) MCANnENDN — Endian Register

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $004_H$

**Value after reset:** $8765\ 4321_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETV[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETV[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.15 MCANnENDN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ETV[31:0] | Endianness Test Value<br>The endianness test value is $87654321_H$. |

ISO CANFD ISO 11898-1:2015

### (3) MCANnDBTP — Data Bit Timing & Prescaler Register

This register is only writable if bits MCANnCCCR.CCE and MCANnCCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 m_can_cclk (CLKP_H2) periods. tq = (DBRP + 1) mtq.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [DTSEG1 + DTSEG2 + 3] tq
or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $00C_H$

**Value after reset:** $0000\ 0A33_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TDC | — | — | DBRP[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | RP | R | R | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DTSEG1[4:0] | | | | | DTSEG2[3:0] | | | | DSJW[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

ISO CAN FD ISO 11898-1:2015

**Table 19.16 MCANnDBTP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. |
| 23 | TDC | Transmitter Delay Compensation<br>0: Transmitter Delay Compensation disabled<br>1: Transmitter Delay Compensation enabled |
| 22, 21 | Reserved | These bits are always read as 0. |
| 20 to 16 | DBRP[4:0] | Data Bit Rate Prescaler<br>$00_H$ to $1F_H$<br>The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Bit Rate Prescaler are 0 to 31. When TDC = 1, the range is limited to 0, 1. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 15 to 13 | Reserved | These bits are always read as 0. |
| 12 to 8 | DTSEG1[4:0] | Data time segment before sample point<br>$00_H$ to $1F_H$<br>Valid values are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 7 to 4 | DTSEG2[3:0] | Data time segment after sample point<br>$0_H$ to $F_H$<br>Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 3 to 0 | DSJW[3:0] | Data (Re) Synchronization Jump Width<br>$0_H$ to $F_H$<br>Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |

**NOTES**

1. With a CAN clock (m_can_cclk: CLKP_H2) of 8 MHz, the reset value of 0000 $0A33_H$ configures the M_CAN for a fast bit rate of 500 kBit/s.
2. The bit rate configured for the CAN FD data phase via MCANnDBTP must be higher or equal to the bit rate configured for the arbitration phase via MCANnNBTP.

ISO CANFD ISO 11898-1:2015

### (4) MCANnTEST — Test Register

Write access to the Test Register has to be enabled by setting bit MCANnCCCR.TEST to '1'. All Test Register functions are set to their reset values when bit MCANnCCCR.TEST is reset.

Loop Back Mode and software control of pin m_can_tx are hardware test modes. Programming of TX ≠ "00" may disturb the message transfer on the CAN bus.

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $010_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RX | TX[1:0] | | LBCK | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | RP | RP | RP | R | R | R | R |

**Table 19.17 MCANnTEST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | These bits are always read as 0. |
| 7 | RX | Receive Pin<br>Monitors the actual value of pin m_can_rx<br>0: The CAN bus is dominant (m_can_rx = '0').<br>1: The CAN bus is recessive (m_can_rx = '1'). |
| 6, 5 | TX[1:0] | Control of Transmit Pin<br>00: Reset value, m_can_tx controlled by the CAN Core, updated at the end of the CAN bit time.<br>01: Sample Point can be monitored at pin m_can_tx.<br>10: Dominant ('0') level at pin m_can_tx<br>11: Recessive ('1') at pin m_can_tx |
| 4 | LBCK | Loop Back Mode<br>0: Reset value, Loop Back Mode is disabled.<br>1: Loop Back Mode is enabled (see **(9) Test Modes**). |
| 3 to 0 | Reserved | These bits are always read as 0. |

ISO CANFD 11898-1:2015

### (5) MCANnRWD — RAM Watchdog

The RAM Watchdog monitors the readiness of the Message RAM. A Message RAM access via the M_CAN's Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by MCANnRWD.WDC. The counter is reloaded with MCANnRWD.WDC when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag MCANnIR.WDI is set. The RAM Watchdog Counter is clocked by the Host clock (m_can_hclk: CLK_HSB).

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $014_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WDV[7:0] | | | | | | | | WDC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP |

**Table 19.18 MCANnRWD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 8 | WDV[7:0] | Watchdog Value<br>Actual Message RAM Watchdog Counter Value. |
| 7 to 0 | WDC[7:0] | Watchdog Configuration<br>Start value of the Message RAM Watchdog Counter. With the reset value of "00" the counter is disabled. |

ISO CANFD CO 11898-1:2015

### (6) MCANnCCCR — CC Control Register

For details about setting and resetting of single bits see **(1) Software Initialization**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $018_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NISO | TXP | EFBI | PXHD | — | — | BRSE | FDOE | TEST | DAR | MON | CSR | CSA | ASM | CCE | INIT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | RP | RP | RP | RP | R | R | RP | RP | Rp | RP | Rp | R/W | R | Rp | RP | R/W |

**Table 19.19 MCANnCCCR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 | NISO | Non ISO Operation<br>If this bit is set, the M_CAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.<br>0: CAN FD frame format according to ISO11898-1:2015<br>1: CAN FD frame format according to Bosch CAN FD Specification V1.0 |
| 14 | TXP | Transmit Pause<br>If this bit is set, the M_CAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame (see **Section 19.6.3.5, Tx Handling**).<br>0: Transmit pause disabled<br>1: Transmit pause enabled |
| 13 | EFBI | Edge Filtering during Bus Integration<br>0: Edge filtering disabled<br>1: Two consecutive dominant tq required to detect an edge for hard synchronization. |
| 12 | PXHD | Protocol Exception Handling Disable<br>0: Protocol exception handling enabled<br>1: Protocol exception handling disabled<br>**NOTE**<br>When protocol exception handling is disabled, the M_CAN will transmit an error frame when it detects a protocol exception condition. |
| 11, 10 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 9 | BRSE | Bit Rate Switch Enable<br>0: Bit rate switching for transmissions disabled.<br>1: Bit rate switching for transmissions enabled.<br>**NOTE**<br>When CAN FD operation is disabled FDOE = '0', BRSE is not evaluated. |
| 8 | FDOE | FD Operation Enable<br>0: FD operation disabled<br>1: FD operation enabled |

ISO CANFD 16845-1:2015

**Table 19.19 MCANnCCCR Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | TEST | Test Mode Enable<br>0: Normal operation, register MCANnTEST holds reset values.<br>1: Test Mode, write access to register MCANnTEST enabled. |
| 6 | DAR | Disable Automatic Retransmission<br>0: Automatic retransmission of messages not transmitted successfully enabled.<br>1: Automatic retransmission disabled |
| 5 | MON | Bus Monitoring Mode<br>Bit MON can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time.<br>0: Bus Monitoring Mode is disabled<br>1: Bus Monitoring Mode is enabled |
| 4 | CSR | Clock Stop Request<br>0: No clock stop is requested.<br>1: Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle. |
| 3 | CSA | Clock Stop Acknowledge<br>0: No clock stop acknowledged.<br>1: M_TTCAN may be set in power down by stopping m_can_hclk (CLK_HSB) and m_can_cclk (CLKP_H2). |
| 2 | ASM | Restricted Operation Mode<br>Bit ASM can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. For a description of the Restricted Operation Mode see **(5) Restricted Operation Mode**.<br>0: Normal CAN operation<br>1: Restricted Operation Mode active |
| 1 | CCE | Configuration Change Enable<br>0: The CPU has no write access to the protected configuration registers.<br>1: The CPU has write access to the protected configuration registers (while MCANnCCCR.INIT = '1'). |
| 0 | INIT | Initialization<br>0: Normal Operation<br>1: Initialization is started. |

**NOTE**

Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

ISO CANFD ISO 11898-1:2015

### (7) MCANnNBTP — Nominal Bit Timing & Prescaler Register

This register is only writable if bits MCANnCCCR.CCE and MCANnCCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 385 time quanta. The CAN time quantum may be programmed in the range of 1 to 512 m_can_cclk (CLKP_H2) periods. tq = (NBRP + 1) mtq.

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [NTSEG1 + NTSEG2 + 3] tq
or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $01C_H$

**Value after reset:** $0600\ 0A03_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NSJW[6:0] | | | | | | | NBRP[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NTSEG1[7:0] | | | | | | | | — | NTSEG2[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

ISO CD IPD ISO 11898-1:2015

**Table 19.20 MCANnNBTP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 25 | NSJW[6:0] | Nominal (Re)Synchronization Jump Width.<br>$00_H$ to $7F_H$<br>Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 24 to 16 | NBRP[8:0] | Nominal Bit Rate Prescaler<br>$000_H$ to $1FF_H$<br>The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Bit Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 15 to 8 | NTSEG1[7:0] | Nominal Time segment before sample point<br>$01_H$ to $FF_H$<br>Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 7 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 6 to 0 | NTSEG2[6:0] | Nominal Time segment after sample point<br>$00_H$ to $7F_H$<br>Valid values are 1 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |

**NOTE**

With a CAN clock (m_can_cclk: CLKP_H2) of 8 MHz, the reset value of $0600\ 0A03_H$ configures the M_CAN for a bit rate of 500 kBit/s.

### (8) MCANnTSCC — Timestamp Counter Configuration

For a description of the Timestamp Counter see **Section 19.5.3.2, Timestamp Generation**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $020_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TCP[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TSS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP |

**Table 19.21 MCANnTSCC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 19 to 16 | TCP[3:0] | Timestamp Counter Prescaler<br>$0_H$ to $F_H$<br>Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 to 16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.<br>**NOTE**<br>With CAN FD an external counter is required for timestamp generation (TSS = “10”) |
| 15 to 2 | Reserved | These bits are always read as 0. |
| 1, 0 | TSS[1:0] | Timestamp Select<br>00: Timestamp counter value always $0000_H$<br>01: Timestamp counter value incremented according to TCP<br>10: External timestamp counter value used<br>11: Same as “00” |

ISO CANFD 11898-1:2015

### (9) MCANnTSCV — Timestamp Counter Value

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $024_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSC[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

**Table 19.22 MCANnTSCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 0 | TSC[15:0] | Timestamp Counter<br>The internal/external Timestamp Counter value is captured on start of frame (both Rx and Tx). When MCANnTSCC.TSS = “01”, the Timestamp Counter is incremented in multiples of CAN bit times [1 to 16] depending on the configuration of MCANnTSCC.TCP. A wrap around sets interrupt flag MCANnIR.TSW. Write access resets the counter to zero. When MCANnTSCC.TSS = “10”, TSC reflects the external Timestamp Counter value. A write access has no impact. |

**NOTE**

A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by write access to MCANnTSCV.

ISO CANFD IP098 - 2015

### (10) MCANnTOCC — Timeout Counter Configuration

For a description of the Timeout Counter see **Section 19.5.3.3, Timeout Counter**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $028_H$

**Value after reset:** FFFF $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TOP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TOS[1:0] | | ETOC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP |

**Table 19.23 MCANnTOCC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TOP[15:0] | Timeout Period<br>Start value of the Timeout Counter (down-counter). Configures the Timeout Period. |
| 15 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2, 1 | TOS[1:0] | Timeout Select<br>When operating in Continuous mode, a write to MCANnTOCV presets the counter to the value configured by MCANnTOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by MCANnTOCC.TOP. Down-counting is started when the first FIFO element is stored.<br>00: Continuous operation<br>01: Timeout controlled by Tx Event FIFO<br>10: Timeout controlled by Rx FIFO 0<br>11: Timeout controlled by Rx FIFO 1 |
| 0 | ETOC | Enable Timeout Counter<br>0: Timeout Counter disabled<br>1: Timeout Counter enabled |

**NOTE**

For use of timeout function with CAN FD see **Section 19.6.3.3, Timeout Counter**.

ISO CANFD ISO 11898-1:2015

### (11) MCANnTOCV — Timeout Counter Value

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $02C_H$

**Value after reset:** $0000\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TOC[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

**Table 19.24 MCANnTOCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 0 | TOC[15:0] | Timeout Counter<br>The Timeout Counter is decremented in multiples of CAN bit times [1 to 16] depending on the configuration of MCANnTSCC.TCP. When decremented to zero, interrupt flag MCANnIR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via MCANnTOCC.TOS. |

ISO CANFD ISO 11898-1:2015

### (12) MCANnECR — Error Counter Register

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $040_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CEL[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | X | X | X | X | X | X | X | X |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RP | REC[6:0] | | | | | | | TEC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.25 MCANnECR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. |
| 23 to 16 | CEL[7:0] | CAN Error Logging<br>The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at $FF_H$; the next increment of TEC or REC sets interrupt flag MCANnIR.ELO. |
| 15 | RP | Receive Error Passive<br>0: The Receive Error Counter is below the error passive level of 128<br>1: The Receive Error Counter has reached the error passive level of 128 |
| 14 to 8 | REC[6:0] | Receive Error Counter<br>Actual state of the Receive Error Counter, values between 0 and 127 |
| 7 to 0 | TEC[7:0] | Transmit Error Counter<br>Actual state of the Transmit Error Counter, values between 0 and 255 |

**NOTE**

When MCANnCCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

ISO CANFD ISO 11898-1:2015

### (13) MCANnPSR — Protocol Status Register

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $044_H$

**Value after reset:** $0000\ 0707_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | TDCV[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | PXE | RFDF | RBRS | RESI | DLEC[2:0] | | | BO | EW | EP | ACT[1:0] | | LEC[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| R/W | R | X | X | X | X | S | S | S | R | R | R | R | R | S | S | S |

**Table 19.26 MCANnPSR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | These bits are always read as 0. |
| 22 to 16 | TDCV[6:0] | Transmitter Delay Compensation Value<br>$00_H$ to $7F_H$<br>Position of the secondary sample point, defined by the sum of the measured delay from m_can_tx to m_can_rx and MCANnTDCR.TDCO. The SSP position is, in the data phase, the number of mtq between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq. |
| 15 | Reserved | These bits are always read as 0. |
| 14 | PXE | Protocol Exception Event<br>0: No protocol exception event occurred since last read access<br>1: Protocol exception event occurred |
| 13 | RFDF | Received a CAN FD Message<br>This bit is set independent of acceptance filtering.<br>0: Since this bit was reset by the CPU, no CAN FD message has been received<br>1: Message in CAN FD format with FDF flag set has been received |
| 12 | RBRS | BRS flag of last received CAN FD Message<br>This bit is set together with RFDF, independent of acceptance filtering.<br>0: Last received CAN FD message did not have its BRS flag set<br>1: Last received CAN FD message had its BRS flag set |
| 11 | RESI | ESI flag of last received CAN FD Message<br>This bit is set together with RFDF, independent of acceptance filtering.<br>0: Last received CAN FD message did not have its ESI flag set<br>1: Last received CAN FD message had its ESI flag set |
| 10 to 8 | DLEC[2:0] | Data Phase Last Error Code<br>Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. |
| 7 | BO | Bus_Off Status<br>0: The M_CAN is not Bus_Off<br>1: The M_CAN is in Bus_Off state |
| 6 | EW | Warning Status<br>0: Both error counters are below the Error_Warning limit of 96<br>1: At least one of error counter has reached the Error_Warning limit of 96 |

ISO CANFD ISO 11898-1:2015

**Table 19.26 MCANnPSR Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 5 | EP | Error Passive<br>0: The M_CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected<br>1: The M_CAN is in the Error_Passive state |
| 4, 3 | ACT[1:0] | Activity<br>Monitors the module's CAN communication state.<br>00: Synchronizing - node is synchronizing on CAN communication<br>01: Idle - node is neither receiver nor transmitter<br>10: Receiver - node is operating as receiver<br>11: Transmitter - node is operating as transmitter<br>**NOTE**<br>ACT is set to "00" by a Protocol Exception Event. |
| 2 to 0 | LEC[2:0] | Last Error Code<br>The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error.<br>0: No Error: No error occurred since LEC has been reset by successful reception or transmission.<br>1: Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.<br>2: Form Error: A fixed format part of a received frame has the wrong format.<br>3: AckError: The message transmitted by the M_TTCAN was not acknowledged by another node.<br>4: Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.<br>5: Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).<br>6: CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.<br>7: NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register. |

ISO CANFD 11898-1:2015

**NOTES**

1. When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in DLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.
2. The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1:2015) cannot be shortened by setting or resetting MCANnCCCR.INIT. If the device goes Bus_Off, it will set MCANnCCCR.INIT of its own accord, stopping all bus activities. Once MCANnCCCR.INIT has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of MCANnCCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to MCANnPSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. MCANnECR.REC is used to count these sequences.

### (14) MCANnTDCR — Transmitter Delay Compensation Register

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $048_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TDCO[6:0] | | | | | | | — | TDCF[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

**Table 19.27 MCANnTDCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 15 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 14 to 8 | TDCO[6:0] | Transmitter Delay Compensation Offset<br>$00_H$ to $7F_H$<br>Offset value defining the distance between the measured delay from m_can_tx to m_can_rx and the secondary sample point. Valid values are 0 to 127 mtq. |
| 7 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 10 to 8 | TDCF[6:0] | Transmitter Delay Compensation Filter Window Length<br>$00_H$ to $7F_H$<br>Defines the minimum value for the SSP position, dominant edges on m_can_rx that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCO. Valid values are 0 to 127 mtq. |

ISO CANFD ISO 11898-1:2015

### (15) MCANnIR — Interrupt Register

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. The configuration of MCANnIE controls whether an interrupt is generated. The configuration of MCANnILS controls on which interrupt line an interrupt is signalled.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $050_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | ARA | PED | PEA | WDI | BO | EW | EP | ELO | BEU | BEC | DRX | TOO | MRAF | TSW |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFL | TEFF | TEFW | TEFN | TFE | TCF | TC | HPM | RF1L | RF1F | RF1W | RF1N | RF0L | RF0F | RF0W | RF0N |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

ISO CANFD IP 11898-1:2015

**Table 19.28 MCANnIR Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 29 | ARA | Access to Reserved Address<br>0: No access to reserved address occurred<br>1: Access to reserved address occurred |
| 28 | PED | Protocol Error in Data Phase (Data Bit Time is used)<br>0: No protocol error in data phase<br>1: Protocol error in data phase detected (MCANnPSR.DLEC ≠ 0,7) |
| 27 | PEA | Protocol Error in Arbitration Phase (Nominal Bit Time is used)<br>0: No protocol error in arbitration phase<br>1: Protocol error in arbitration phase detected (MCANnPSR.LEC ≠ 0,7) |
| 26 | WDI | Watchdog Interrupt<br>0: No Message RAM Watchdog event occurred<br>1: Message RAM Watchdog event due to missing READY |
| 25 | BO | Bus_Off Status<br>0: Bus_Off status unchanged<br>1: Bus_Off status changed |
| 24 | EW | Warning Status<br>0: Error_Warning status unchanged<br>1: Error_Warning status changed |
| 23 | EP | Error Passive<br>0: Error_Passive status unchanged<br>1: Error_Passive status changed |
| 22 | ELO | Error Logging Overflow<br>0: CAN Error Logging Counter did not overflow<br>1: Overflow of CAN Error Logging Counter occurred |

Table 19.28 MCANnIR Register Contents (2/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 21 | BEU | Bit Error Uncorrected<br>Message RAM bit error detected, uncorrected. Controlled by input signal m_can_aeim_berr[1] generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets MCANnCCCR.INIT to '1'. This is done to avoid transmission of corrupted data.<br>0: No bit error detected when reading from Message RAM<br>1: Bit error detected, uncorrected (e.g. parity logic) |
| 20 | BEC | Bit Error Corrected<br>Message RAM bit error detected and corrected. Controlled by input signal m_can_aeim_berr[0] generated by an optional external parity / ECC logic attached to the Message RAM.<br>0: No bit error detected when reading from Message RAM<br>1: Bit error detected and corrected (e.g. ECC) |
| 19 | DRX | Message stored to Dedicated Rx Buffer<br>The flag is set whenever a received message has been stored into a dedicated Rx Buffer.<br>0: No Rx Buffer updated<br>1: At least one received message stored into an Rx Buffer |
| 18 | TOO | Timeout Occurred<br>0: No timeout<br>1: Timeout reached |
| 17 | MRAF | Message RAM Access Failure<br>The flag is set, when the Rx Handler<br>• has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message.<br>• was not able to write a message to the Message RAM. In this case message storage is aborted.<br>In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location.<br>The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the M_CAN is switched into Restricted Operation Mode (see **(5) Restricted Operation Mode**). To leave Restricted Operation Mode, the Host CPU has to reset MCANnCCCR.ASM.<br>0: No Message RAM access failure occurred<br>1: Message RAM access failure occurred |
| 16 | TSW | Timestamp Wraparound<br>0: No timestamp counter wrap-around<br>1: Timestamp counter wrapped around |
| 15 | TEFL | Tx Event FIFO Element Lost<br>0: No Tx Event FIFO element lost<br>1: Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero |
| 14 | TEFF | Tx Event FIFO Full<br>0: Tx Event FIFO not full<br>1: Tx Event FIFO full |
| 13 | TEFW | Tx Event FIFO Watermark Reached<br>0: Tx Event FIFO fill level below watermark<br>1: Tx Event FIFO fill level reached watermark |
| 12 | TEFN | Tx Event FIFO New Entry<br>0: Tx Event FIFO unchanged<br>1: Tx Handler wrote Tx Event FIFO element |
| 11 | TFE | Tx FIFO Empty<br>0: Tx FIFO non-empty<br>1: Tx FIFO empty |

ISO CANFD ISO 11898-1:2015

Table 19.28 MCANnIR Register Contents (3/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 10 | TCF | Transmission Cancellation Finished<br>0: No transmission cancellation finished<br>1: Transmission cancellation finished |
| 9 | TC | Transmission Completed<br>0: No transmission completed<br>1: Transmission completed |
| 8 | HPM | High Priority Message<br>0: No high priority message received<br>1: High priority message received |
| 7 | RF1L | Rx FIFO 1 Message Lost<br>0: No Rx FIFO 1 message lost<br>1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero |
| 6 | RF1F | Rx FIFO 1 Full<br>0: Rx FIFO 1 not full<br>1: Rx FIFO 1 full |
| 5 | RF1W | Rx FIFO 1 Watermark Reached<br>0: Rx FIFO 1 fill level below watermark<br>1: Rx FIFO 1 fill level reached watermark |
| 4 | RF1N | Rx FIFO 1 New Message<br>0: No new message written to Rx FIFO 1<br>1: New message written to Rx FIFO 1 |
| 3 | RF0L | Rx FIFO 0 Message Lost<br>0: No Rx FIFO 0 message lost<br>1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero |
| 2 | RF0F | Rx FIFO 0 Full<br>0: Rx FIFO 0 not full<br>1: Rx FIFO 0 full |
| 1 | RF0W | Rx FIFO 0 Watermark Reached<br>0: Rx FIFO 0 fill level below watermark<br>1: Rx FIFO 0 fill level reached watermark |
| 0 | RF0N | Rx FIFO 0 New Message<br>0: No new message written to Rx FIFO 0<br>1: New message written to Rx FIFO 0 |

ISO CANFD CC (1889-10-2015

### (16) MCANnIE — Interrupt Enable

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signalled on an interrupt line.

0: Interrupt disabled
1: Interrupt enabled

**Access:** This register can be read/written in 32-bit units.
**Address:** <MCANn_base> + $054_H$
**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | ARAE | PEDE | PEAE | WDIE | BOE | EWE | EPE | ELOE | BEUE | BECE | DRXE | TOOE | MRAFE | TSWE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFLE | TEFFE | TEFWE | TEFNE | TFEE | TCFE | TCE | HPME | RF1LE | RF1FE | RF1WE | RF1NE | RF0LE | RF0FE | RF0WE | RF0NE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

ISO CANFD IDS - 11898-1:2015

**Table 19.29 MCANnIE Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 29 | ARAE | Access to Reserved Address Enable |
| 28 | PEDE | Protocol Error in Data Phase Enable |
| 27 | PEAE | Protocol Error in Arbitration Phase Enable |
| 26 | WDIE | Watchdog Interrupt Enable |
| 25 | BOE | Bus_Off Status Interrupt Enable |
| 24 | EWE | Warning Status Interrupt Enable |
| 23 | EPE | Error Passive Interrupt Enable |
| 22 | ELOE | Error Logging Overflow Interrupt Enable |
| 21 | BEUE | Bit Error Uncorrected Interrupt Enable |
| 20 | BECE | Bit Error Corrected Interrupt Enable |
| 19 | DRXE | Message stored to Dedicated Rx Buffer Interrupt Enable |
| 18 | TOOE | Timeout Occurred Interrupt Enable |
| 17 | MRAFE | Message RAM Access Failure Interrupt Enable |
| 16 | TSWE | Timestamp Wraparound Interrupt Enable |
| 15 | TEFLE | Tx Event FIFO Event Lost Interrupt Enable |
| 14 | TEFFE | Tx Event FIFO Full Interrupt Enable |
| 13 | TEFWE | Tx Event FIFO Watermark Reached Interrupt Enable |
| 12 | TEFNE | Tx Event FIFO New Entry Interrupt Enable |
| 11 | TFEE | Tx FIFO Empty Interrupt Enable |
| 10 | TCFE | Transmission Cancellation Finished Interrupt Enable |
| 9 | TCE | Transmission Completed Interrupt Enable |
| 8 | HPME | High Priority Message Interrupt Enable |
| 7 | RF1LE | Rx FIFO 1 Message Lost Interrupt Enable |

Table 19.29 MCANnIE Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 | RF1FE | Rx FIFO 1 Full Interrupt Enable |
| 5 | RF1WE | Rx FIFO 1 Watermark Reached Interrupt Enable |
| 4 | RF1NE | Rx FIFO 1 New Message Interrupt Enable |
| 3 | RF0LE | Rx FIFO 0 Message Lost Interrupt Enable |
| 2 | RF0FE | Rx FIFO 0 Full Interrupt Enable |
| 1 | RF0WE | Rx FIFO 0 Watermark Reached Interrupt Enable |
| 0 | RF0NE | Rx FIFO 0 New Message Interrupt Enable |

ISO CANFD ISO 11898-1:2015

### (17) MCANnILS — Interrupt Line Select

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via MCANnILE.EINT0 and MCANnILE.EINT1.

0: Interrupt assigned to interrupt line m_can_int0 (INTMCANnI0)
1: Interrupt assigned to interrupt line m_can_int1 (INTMCANnI1)

**Access:** This register can be read/written in 32-bit units.
**Address:** <MCANn_base> + $058_H$
**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | ARAL | PEDL | PEAL | WDIL | BOL | EWL | EPL | ELOL | BEUL | BECL | DRXL | TOOL | MRAFL | TSWL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFLL | TEFFL | TEFWL | TEFNL | TFEL | TCFL | TCL | HPML | RF1LL | RF1FL | RF1WL | RF1NL | RF0LL | RF0FL | RF0WL | RF0NL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

ISO CANFD IP 11898-1:2015

**Table 19.30 MCANnILS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 29 | ARAL | Access to Reserved Address Line |
| 28 | PEDL | Protocol Error in Data Phase Line |
| 27 | PEAL | Protocol Error in Arbitration Phase Line |
| 26 | WDIL | Watchdog Interrupt Line |
| 25 | BOL | Bus_Off Status Interrupt Line |
| 24 | EWL | Warning Status Interrupt Line |
| 23 | EPL | Error Passive Interrupt Line |
| 22 | ELOL | Error Logging Overflow Interrupt Line |
| 21 | BEUL | Bit Error Uncorrected Interrupt Line |
| 20 | BECL | Bit Error Corrected Interrupt Line |
| 19 | DRXL | Message stored to Dedicated Rx Buffer Interrupt Line |
| 18 | TOOL | Timeout Occurred Interrupt Line |
| 17 | MRAFL | Message RAM Access Failure Interrupt Line |
| 16 | TSWL | Timestamp Wraparound Interrupt Line |
| 15 | TEFLL | Tx Event FIFO Event Lost Interrupt Line |
| 14 | TEFFL | Tx Event FIFO Full Interrupt Line |
| 13 | TEFWL | Tx Event FIFO Watermark Reached Interrupt Line |
| 12 | TEFNL | Tx Event FIFO New Entry Interrupt Line |
| 11 | TFEL | Tx FIFO Empty Interrupt Line |
| 10 | TCFL | Transmission Cancellation Finished Interrupt Line |
| 9 | TCL | Transmission Completed Interrupt Line |
| 8 | HPML | High Priority Message Interrupt Line |

Table 19.30 MCANnILS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RF1LL | Rx FIFO 1 Message Lost Interrupt Line |
| 6 | RF1FL | Rx FIFO 1 Full Interrupt Line |
| 5 | RF1WL | Rx FIFO 1 Watermark Reached Interrupt Line |
| 4 | RF1NL | Rx FIFO 1 New Message Interrupt Line |
| 3 | RF0LL | Rx FIFO 0 Message Lost Interrupt Line |
| 2 | RF0FL | Rx FIFO 0 Full Interrupt Line |
| 1 | RF0WL | Rx FIFO 0 Watermark Reached Interrupt Line |
| 0 | RF0NL | Rx FIFO 0 New Message Interrupt Line |

ISO CANFD ISO 11898-1:2015

### (18) MCANnILE — Interrupt Line Enable

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $05C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EINT1 | EINT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 19.31 MCANnILE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1 | EINT1 | Enable Interrupt Line 1<br>0: Interrupt line m_can_int1 (INTMCANnI1) disabled<br>1: Interrupt line m_can_int1 (INTMCANnI1) enabled |
| 0 | EINT0 | Enable Interrupt Line 0<br>0: Interrupt line m_can_int0 (INTMCANnI0) disabled<br>1: Interrupt line m_can_int0 (INTMCANnI0) enabled |

ISO CANFD ISO 11898-1:2015

### (19) MCANnGFC — Global Filter Configuration

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in **Figure 19.8, Standard Message ID Filter Path** and **Figure 19.9, Extended Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $080_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ANFS[1:0] | | ANFE[1:0] | | RRFS | RRFE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP |

**Table 19.32 MCANnGFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5, 4 | ANFS[1:0] | Accept Non-matching Frames Standard<br>Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.<br>00: Accept in Rx FIFO 0<br>01: Accept in Rx FIFO 1<br>10: Reject<br>11: Reject |
| 3, 2 | ANFE[1:0] | Accept Non-matching Frames Extended<br>Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.<br>00: Accept in Rx FIFO 0<br>01: Accept in Rx FIFO 1<br>10: Reject<br>11: Reject |
| 1 | RRFS | Reject Remote Frames Standard<br>0: Filter remote frames with 11-bit standard IDs<br>1: Reject all remote frames with 11-bit standard IDs |
| 0 | RRFE | Reject Remote Frames Extended<br>0: Filter remote frames with 29-bit extended IDs<br>1: Reject all remote frames with 29-bit extended IDs |

ISO CANFD ISO 11898-1:2015

### (20) MCANnSIDFC — Standard ID Filter Configuration

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in **Figure 19.8, Standard Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $084_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | LSS[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLSSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.33 MCANnSIDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 16 | LSS[7:0] | List Size Standard<br>0: No standard Message ID filter<br>1 to 128: Number of standard Message ID filter elements<br>>128: Values greater than 128 are interpreted as 128 |
| 15 to 2 | FLSSA[15:2] | Filter List Standard Start Address<br>Start address of standard Message ID filter list (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

ISO CANFD ISO 11898-1:2015

### (21) MCANnXIDFC — Extended ID Filter Configuration

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in **Figure 19.9, Extended Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $088_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | LSE[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLESA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.34 MCANnXIDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 22 to 16 | LSE[6:0] | List Size Extended<br>0: No extended Message ID filter<br>1 to 64: Number of extended Message ID filter elements<br>>64: Values greater than 64 are interpreted as 64 |
| 15 to 2 | FLESA[15:2] | Filter List Extended Start Address<br>Start address of extended Message ID filter list (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

ISO CANFD ISO 11898-1:2015

### (22) MCANnXIDAM — Extended ID AND Mask

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $090_H$

**Value after reset:** $1FFF\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | EIDM[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EIDM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

**Table 19.35 MCANnXIDAM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 28 to 0 | EIDM[28:0] | Extended ID Mask<br>For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active. |

ISO CANFD ISO 11898-1:2015

### (23) MCANnHPMS — High Priority Message Status

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $094_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLST | FIDX[6:0] | | | | | | | MSI[1:0] | | BIDX[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.36 MCANnHPMS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. |
| 15 | FLST | Filter List<br>Indicates the filter list of the matching filter element.<br>0: Standard Filter List<br>1: Extended Filter List |
| 14 to 8 | FIDX[6:0] | Filter Index<br>Index of matching filter element. Range is 0 to MCANnSIDFC.LSS – 1 resp. MCANnXIDFC.LSE – 1. |
| 7, 6 | MSI[1:0] | Message Storage Indicator<br>00: No FIFO selected<br>01: FIFO message lost<br>10: Message stored in FIFO 0<br>11: Message stored in FIFO 1 |
| 5 to 0 | BIDX[5:0] | Buffer Index<br>Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = ‘1’. |

ISO CANFD SOT 898-1:2015

### (24) MCANnNDAT1 — New Data 1

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $098_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND31 | ND30 | ND29 | ND28 | ND27 | ND26 | ND25 | ND24 | ND23 | ND22 | ND21 | ND20 | ND19 | ND18 | ND17 | ND16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND15 | ND14 | ND13 | ND12 | ND11 | ND10 | ND9 | ND8 | ND7 | ND6 | ND5 | ND4 | ND3 | ND2 | ND1 | ND0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.37 MCANnNDAT1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ND[31:0] | New Data<br>The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.<br>0: Rx Buffer not updated<br>1: Rx Buffer updated from new message |

ISO CANFD ISO 11898-1:2015

### (25) MCANnNDAT2 — New Data 2

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $09C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND63 | ND62 | ND61 | ND60 | ND59 | ND58 | ND57 | ND56 | ND55 | ND54 | ND53 | ND52 | ND51 | ND50 | ND49 | ND48 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND47 | ND46 | ND45 | ND44 | ND43 | ND42 | ND41 | ND40 | ND39 | ND38 | ND37 | ND36 | ND35 | ND34 | ND33 | ND32 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.38 MCANnNDAT2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ND[63:32] | New Data<br>The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.<br>0: Rx Buffer not updated<br>1: Rx Buffer updated from new message |

ISO CANFD ISO 11898-1:2015

### (26) MCANnRXF0C — Rx FIFO 0 Configuration

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0A0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F0OM | F0WM[6:0] | | | | | | | — | F0S[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F0SA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.39 MCANnRXF0C Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | F0OM | FIFO 0 Operation Mode<br>FIFO 0 can be operated in blocking or in overwrite mode (see **(2) Rx FIFOs**).<br>0: FIFO 0 blocking mode<br>1: FIFO 0 overwrite mode |
| 30 to 24 | F0WM[6:0] | Rx FIFO 0 Watermark<br>0: Watermark interrupt disabled<br>1 to 64: Level for Rx FIFO 0 watermark interrupt (MCANnIR.RF0W)<br>>64: Watermark interrupt disabled |
| 23 | Reserved | This bit is always read as 0. |
| 22 to 16 | F0S[6:0] | Rx FIFO 0 Size<br>0: No Rx FIFO 0<br>1 to 64: Number of Rx FIFO 0 elements<br>>64: Values greater than 64 are interpreted as 64<br>The Rx FIFO 0 elements are indexed from 0 to F0S-1 |
| 15 to 2 | F0SA[15:2] | Rx FIFO 0 Start Address<br>Start address of Rx FIFO 0 in Message RAM (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. |

ISO CANFD (ISO 11898-1:2015

### (27) MCANnRXF0S — Rx FIFO 0 Status

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0A4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RF0L | F0F | — | — | F0PI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | F0GI[5:0] | | | | | | — | F0FL[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.40 MCANnRXF0S Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. |
| 25 | RF0L | Rx FIFO 0 Message Lost<br>This bit is a copy of interrupt flag MCANnIR.RF0L. When MCANnIR.RF0L is reset, this bit is also reset.<br>0: No Rx FIFO 0 message lost<br>1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero<br>**NOTE**<br>Overwriting the oldest message when MCANnRXF0C.F0OM = '1' will not set this flag. |
| 24 | F0F | Rx FIFO 0 Full<br>0: Rx FIFO 0 not full<br>1: Rx FIFO 0 full |
| 23, 22 | Reserved | These bits are always read as 0. |
| 21 to 16 | F0PI[5:0] | Rx FIFO 0 Put Index<br>Rx FIFO 0 write index pointer, range 0 to 63. |
| 15, 14 | Reserved | These bits are always read as 0. |
| 13 to 8 | F0GI[5:0] | Rx FIFO 0 Get Index<br>Rx FIFO 0 read index pointer, range 0 to 63. |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 0 | F0FL[6:0] | Rx FIFO 0 Fill Level<br>Number of elements stored in Rx FIFO 0, range 0 to 64. |

ISO CANFD ... 2015

### (28) MCANnRXF0A — Rx FIFO 0 Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0A8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | F0AI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.41 MCANnRXF0A Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 to 0 | F0AI[5:0] | Rx FIFO 0 Acknowledge Index<br>After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index MCANnRXF0S.F0GI to F0AI + 1 and update the FIFO 0 Fill Level MCANnRXF0S.F0FL. |

ISO CANFD ISO 11898-1:2015

### (29) MCANnRXBC — Rx Buffer Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0AC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RBSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.42 MCANnRXBC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 2 | RBSA[15:2] | Rx Buffer Start Address<br>Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C. |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

ISO CANFD ISO 11898-1:2015

### (30) MCANnRXF1C — Rx FIFO 1 Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0B0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1OM | F1WM[6:0] | | | | | | | — | F1S[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1SA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.43 MCANnRXF1C Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | F1OM | FIFO 1 Operation Mode<br>FIFO 1 can be operated in blocking or in overwrite mode (see **(2) Rx FIFOs**).<br>0: FIFO 1 blocking mode<br>1: FIFO 1 overwrite mode |
| 30 to 24 | F1WM[6:0] | Rx FIFO 1 Watermark<br>0: Watermark interrupt disabled<br>1 to 64: Level for Rx FIFO 1 watermark interrupt (MCANnIR.RF1W)<br>>64: Watermark interrupt disabled |
| 23 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 22 to 16 | F1S[6:0] | Rx FIFO 1 Size<br>0: No Rx FIFO 1<br>1 to 64: Number of Rx FIFO 1 elements<br>>64: Values greater than 64 are interpreted as 64<br>The Rx FIFO 1 elements are indexed from 0 to F1S - 1 |
| 15 to 2 | F1SA[15:2] | Rx FIFO 1 Start Address<br>Start address of Rx FIFO 1 in Message RAM (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

ISO CANFD (ISO 11898-1:2015

### (31) MCANnRXF1S — Rx FIFO 1 Status

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0B4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMS[1:0] | | — | — | — | — | RF1L | F1F | — | — | F1PI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | F1GI[5:0] | | | | | | — | F1FL[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.44 MCANnRXF1S Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | DMS[1:0] | Debug Message Status<br>00: Idle state, wait for reception of debug messages, DMA request is cleared<br>01: Debug message A received<br>10: Debug messages A, B received<br>11: Debug messages A, B, C received, DMA request is set |
| 29 to 26 | Reserved | These bits are always read as 0. |
| 25 | RF1L | Rx FIFO 1 Message Lost<br>This bit is a copy of interrupt flag MCANnIR.RF1L. When MCANnIR.RF1L is reset, this bit is also reset.<br>0: No Rx FIFO 1 message lost<br>1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero<br>**NOTE**<br>Overwriting the oldest message when MCANnRXF1C.F1OM = '1' will not set this flag. |
| 24 | F1F | Rx FIFO 1 Full<br>0: Rx FIFO 1 not full<br>1: Rx FIFO 1 full |
| 23, 22 | Reserved | These bits are always read as 0. |
| 21 to 16 | F1PI[5:0] | Rx FIFO 1 Put Index<br>Rx FIFO 1 write index pointer, range 0 to 63. |
| 15, 14 | Reserved | These bits are always read as 0. |
| 13 to 8 | F1GI[5:0] | Rx FIFO 1 Get Index<br>Rx FIFO 1 read index pointer, range 0 to 63. |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 0 | F1FL[6:0] | Rx FIFO 1 Fill Level<br>Number of elements stored in Rx FIFO 1, range 0 to 64. |

ISO CANFD IP Core - 2015

### (32) MCANnRXF1A — Rx FIFO 1 Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0B8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | F1AI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.45 MCANnRXF1A Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 to 0 | F1AI[5:0] | Rx FIFO 1 Acknowledge Index<br>After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index MCANnRXF1S.F1GI to F1AI + 1 and update the FIFO 1 Fill Level MCANnRXF1S.F1FL. |

ISO CANFD ISO 11898-1:2015

### (33) MCANnRXESC — Rx Buffer / FIFO Element Size Configuration

Configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0BC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RBDS[2:0] | | | — | F1DS[2:0] | | | — | F0DS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | RP | RP | RP | R | RP | RP | RP | R | RP | RP | RP |

**Table 19.46 MCANnRXESC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | These bits are always read as 0. |
| 10 to 8 | RBDS[2:0] | Rx Buffer Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 4 | F1DS[2:0] | Rx FIFO 1 Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |
| 3 | Reserved | This bit is always read as 0. |
| 2 to 0 | F0DS[2:0] | Rx FIFO 0 Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |

**NOTE**

In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by MCANnRXESC are stored to the Rx Buffer resp. Rx FIFO element. The rest of the frame's data field is ignored.

ISO CANFD 11898-1:2015

### (34) MCANnTXBC — Tx Buffer Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0C0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TFQM | TFQS[5:0] | | | | | | — | — | NDTB[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | RP | RP | RP | RP | RP | RP | RP | R | R | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.47 MCANnTXBC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 30 | TFQM | Tx FIFO/Queue Mode<br>0: Tx FIFO operation<br>1: Tx Queue operation |
| 29 to 24 | TFQS[5:0] | Transmit FIFO/Queue Size<br>0: No Tx FIFO/Queue<br>1 to 32: Number of Tx Buffers used for Tx FIFO/Queue<br>>32: Values greater than 32 are interpreted as 32 |
| 23, 22 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 21 to 16 | NDTB[5:0] | Number of Dedicated Transmit Buffers<br>0: No Dedicated Tx Buffers<br>1 to 32: Number of Dedicated Tx Buffers<br>>32: Values greater than 32 are interpreted as 32 |
| 15 to 2 | TBSA[15:2] | Tx Buffers Start Address<br>Start address of Tx Buffers section in Message RAM (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

**NOTE**

Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

ISO CANFD 11898-1:2015

### (35) MCANnTXFQS — Tx FIFO/Queue Status

The Tx FIFO/Queue status is related to the pending Tx requests listed in register MCANnTXBRP. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (MCANnTXBRP not yet updated).

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0C4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | TFQF | TFQPI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | TFGI[4:0] | | | | | — | — | TFFL[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.48 MCANnTXFQS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | These bits are always read as 0. |
| 21 | TFQF | Tx FIFO/Queue Full<br>0: Tx FIFO/Queue not full<br>1: Tx FIFO/Queue full |
| 20 to 16 | TFQPI[4:0] | Tx FIFO/Queue Put Index<br>Tx FIFO/Queue write index pointer, range 0 to 31. |
| 15 to 13 | Reserved | These bits are always read as 0. |
| 12 to 8 | TFGI[4:0] | Tx FIFO Get Index<br>Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (MCANnTXBC.TFQM = '1'). |
| 7, 6 | Reserved | These bits are always read as 0. |
| 5 to 0 | TFFL[5:0] | Tx FIFO Free Level<br>Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (MCANnTXBC.TFQM = '1') |

**NOTE**

In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers.
Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

ISO CANFD FDIS 11898-1:2015

### (36) MCANnTXESC — Tx Buffer Element Size Configuration

Configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0C8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TBDS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP |

**Table 19.49 MCANnTXESC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 to 0 | TBDS[2:0] | Tx Buffer Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |

**NOTE**

In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size MCANnTXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as "$CC_H$" (padding bytes).

ISO CANFD ISO 11898-1:2015

### (37) MCANnTXBRP — Tx Buffer Request Pending

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0CC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRP31 | TRP30 | TRP29 | TRP28 | TRP27 | TRP26 | TRP25 | TRP24 | TRP23 | TRP22 | TRP21 | TRP20 | TRP19 | TRP18 | TRP17 | TRP16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRP15 | TRP14 | TRP13 | TRP12 | TRP11 | TRP10 | TRP9 | TRP8 | TRP7 | TRP6 | TRP5 | TRP4 | TRP3 | TRP2 | TRP1 | TRP0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.50 MCANnTXBRP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TRP[31:0] | Transmission Request Pending<br>Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register MCANnTXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register MCANnTXBCR.<br>MCANnTXBRP bits are set only for those Tx Buffers configured via MCANnTXBC. After a MCANnTXBRP bit has been set, a Tx scan (see **Section 19.5.3.5, Tx Handling**) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID).<br>A cancellation request resets the corresponding transmission request pending bit of register MCANnTXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding MCANnTXBRP bit has been reset.<br>After a cancellation has been requested, a finished cancellation is signalled via MCANnTXBCF<br>• after successful transmission together with the corresponding MCANnTXBTO bit<br>• when the transmission has not yet been started at the point of cancellation<br>• when the transmission has been aborted due to lost arbitration<br>• when an error occurred during frame transmission<br>In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding MCANnTXBCF bit is set for all unsuccessful transmissions.<br>0: No transmission request pending<br>1: Transmission request pending<br>**NOTE**<br>MCANnTXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding MCANnTXBRP bit is reset. |

ISO CANFD IP Core © 2015

### (38) MCANnTXBAR — Tx Buffer Add Request

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0D0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AR31 | AR30 | AR29 | AR28 | AR27 | AR26 | AR25 | AR24 | AR23 | AR22 | AR21 | AR20 | AR19 | AR18 | AR17 | AR16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AR15 | AR14 | AR13 | AR12 | AR11 | AR10 | AR9 | AR8 | AR7 | AR6 | AR5 | AR4 | AR3 | AR2 | AR1 | AR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.51 MCANnTXBAR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | AR[31:0] | Add Request<br>Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to MCANnTXBAR. MCANnTXBAR bits are set only for those Tx Buffers configured via MCANnTXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.<br>0: No transmission request added<br>1: Transmission requested added |

**NOTE**

If an add request is applied for a Tx Buffer with pending transmission request (corresponding MCANnTXBRP bit already set), this add request is ignored.

ISO CAN FD ISO 11898-1:2015

### (39) MCANnTXBCR — Tx Buffer Cancellation Request

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0D4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CR31 | CR30 | CR29 | CR28 | CR27 | CR26 | CR25 | CR24 | CR23 | CR22 | CR21 | CR20 | CR19 | CR18 | CR17 | CR16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CR15 | CR14 | CR13 | CR12 | CR11 | CR10 | CR9 | CR8 | CR7 | CR6 | CR5 | CR4 | CR3 | CR2 | CR1 | CR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.52 MCANnTXBCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CR[31:0] | Cancellation Request<br>Each Tx Buffer has its own Cancellation Request bit. Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to MCANnTXBCR. MCANnTXBCR bits are set only for those Tx Buffers configured via MCANnTXBC. The bits remain set until the corresponding bit of MCANnTXBRP is reset.<br>0: No cancellation pending<br>1: Cancellation pending |

ISO CANFD ISO 11898-1:2015

### (40) MCANnTXBTO — Tx Buffer Transmission Occurred

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0D8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TO31 | TO30 | TO29 | TO28 | TO27 | TO26 | TO25 | TO24 | TO23 | TO22 | TO21 | TO20 | TO19 | TO18 | TO17 | TO16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TO15 | TO14 | TO13 | TO12 | TO11 | TO10 | TO9 | TO8 | TO7 | TO6 | TO5 | TO4 | TO3 | TO2 | TO1 | TO0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.53 MCANnTXBTO Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TO[31:0] | Transmission Occurred<br>Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding MCANnTXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register MCANnTXBAR.<br>0: No transmission occurred<br>1: Transmission occurred |

ISO CANFD ISO 11898-1:2015

### (41) MCANnTXBCF — Tx Buffer Cancellation Finished

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0DC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF31 | CF30 | CF29 | CF28 | CF27 | CF26 | CF25 | CF24 | CF23 | CF22 | CF21 | CF20 | CF19 | CF18 | CF17 | CF16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF15 | CF14 | CF13 | CF12 | CF11 | CF10 | CF9 | CF8 | CF7 | CF6 | CF5 | CF4 | CF3 | CF2 | CF1 | CF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.54 MCANnTXBCF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CF[31:0] | Cancellation Finished<br>Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding MCANnTXBRP bit is cleared after a cancellation was requested via MCANnTXBCR. In case the corresponding MCANnTXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register MCANnTXBAR.<br>0: No transmit buffer cancellation<br>1: Transmit buffer cancellation finished |

ISO CANFD ISO 11898-1:2015

### (42) MCANnTXBTIE — Tx Buffer Transmission Interrupt Enable

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0E0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIE31 | TIE30 | TIE29 | TIE28 | TIE27 | TIE26 | TIE25 | TIE24 | TIE23 | TIE22 | TIE21 | TIE20 | TIE19 | TIE18 | TIE17 | TIE16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIE15 | TIE14 | TIE13 | TIE12 | TIE11 | TIE10 | TIE9 | TIE8 | TIE7 | TIE6 | TIE5 | TIE4 | TIE3 | TIE2 | TIE1 | TIE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.55 MCANnTXBTIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TIE[31:0] | Transmission Interrupt Enable<br>Each Tx Buffer has its own Transmission Interrupt Enable bit.<br>0: Transmission interrupt disabled<br>1: Transmission interrupt enable |

### (43) MCANnTXBCIE — Tx Buffer Cancellation Finished Interrupt Enable

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0E4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIE31 | CFIE30 | CFIE29 | CFIE28 | CFIE27 | CFIE26 | CFIE25 | CFIE24 | CFIE23 | CFIE22 | CFIE21 | CFIE20 | CFIE19 | CFIE18 | CFIE17 | CFIE16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIE15 | CFIE14 | CFIE13 | CFIE12 | CFIE11 | CFIE10 | CFIE9 | CFIE8 | CFIE7 | CFIE6 | CFIE5 | CFIE4 | CFIE3 | CFIE2 | CFIE1 | CFIE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.56 MCANnTXBCIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CFIE[31:0] | Cancellation Finished Interrupt Enable<br>Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit.<br>0: Cancellation finished interrupt disabled<br>1: Cancellation finished interrupt enabled |

ISO CANFD ISO 11898-1:2015

### (44) MCANnTXEFC — Tx Event FIFO Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0F0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | EFWM[5:0] | | | | | | — | — | EFS[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | RP | RP | RP | RP | RP | RP | R | R | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EFSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.57 MCANnTXEFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 29 to 24 | EFWM[5:0] | Event FIFO Watermark<br>0: Watermark interrupt disabled<br>1 to 32: Level for Tx Event FIFO watermark interrupt (MCANnIR.TEFW)<br>>32: Watermark interrupt disabled |
| 23, 22 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 21 to 16 | EFS[5:0] | Event FIFO Size<br>0: Tx Event FIFO disabled<br>1 to 32: Number of Tx Event FIFO elements<br>>32: Values greater than 32 are interpreted as 32<br>The Tx Event FIFO elements are indexed from 0 to EFS – 1 |
| 15 to 2 | EFSA[15:2] | Event FIFO Start Address<br>Start address of Tx Event FIFO in Message RAM (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

ISO CANFD CC Mode - 2015

### (45) MCANnTXEFS — Tx Event FIFO Status

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0F4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TEFL | EFF | — | — | — | EFPI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | EFGI[4:0] | | | | | — | — | EFFL[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.58 MCANnTXEFS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. |
| 25 | TEFL | Tx Event FIFO Element Lost<br>This bit is a copy of interrupt flag MCANnIR.TEFL. When MCANnIR.TEFL is reset, this bit is also reset.<br>0: No Tx Event FIFO element lost<br>1: Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero. |
| 24 | EFF | Event FIFO Full<br>0: Tx Event FIFO not full<br>1: Tx Event FIFO full |
| 23 to 21 | Reserved | These bits are always read as 0. |
| 20 to 16 | EFPI[4:0] | Event FIFO Put Index<br>Tx Event FIFO write index pointer, range 0 to 31. |
| 15 to 13 | Reserved | These bits are always read as 0. |
| 12 to 8 | EFGI[4:0] | Event FIFO Get Index<br>Tx Event FIFO read index pointer, range 0 to 31. |
| 7, 6 | Reserved | These bits are always read as 0. |
| 5 to 0 | EFFL[5:0] | Event FIFO Fill Level<br>Number of elements stored in Tx Event FIFO, range 0 to 32. |

ISO CANFD ISO 11898-1:2015

### (46) MCANnTXEFA — Tx Event FIFO Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0F8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | EFAI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 19.59 MCANnTXEFA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 4 to 0 | EFAI[4:0] | Event FIFO Acknowledge Index<br>After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index MCANnTXEFS.EFGI to EFAI + 1 and update the FIFO Fill Level MCANnTXEFS.EFFL. |

ISO CANFD ISO 11898-1:2015

### 19.5.2.4 Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration a single- or dual-ported Message RAM has to be connected to the M_CAN module.

#### (1) Message RAM Configuration

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_CAN module can be configured to allocate up to 4352 words in the Message RAM. It is not necessary to configure each of the sections listed in **Figure 19.4, Message RAM Configuration**, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via MCANnRXESC.F0DS, MCANnRXESC.F1DS, MCANnRXESC.RBDS, and MCANnTXESC.TBDS.

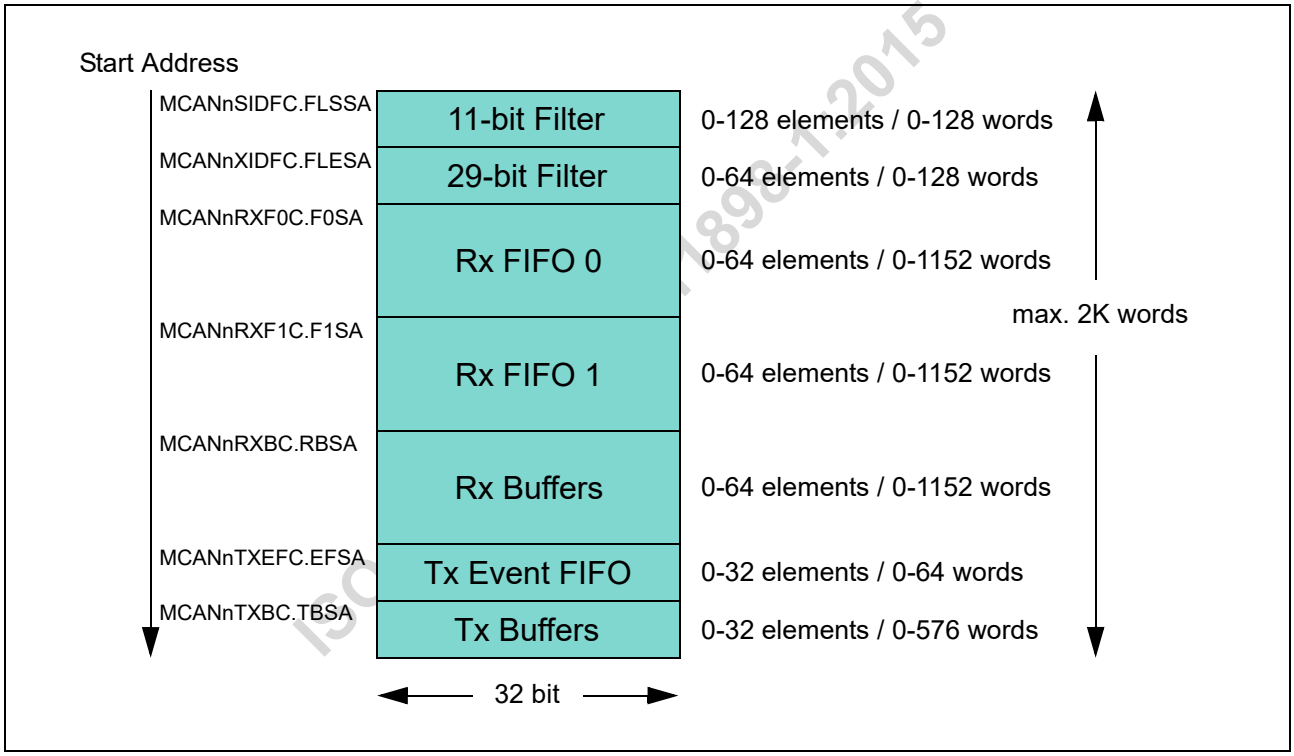



**Figure 19.4 Message RAM Configuration**

When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

**NOTE**

The M_CAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

### (2) Rx Buffer and FIFO Element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in **Table 19.60** below. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register MCANnRXESC.

**Table 19.60 Rx Buffer and FIFO Element**

| | 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|---|
| R0 | ESI, XTD, RTR, ID[28:0] | | | |
| R1 | ANMF, FIDX[6:0] | res, FDF, BRS, DLC[3:0] | RXTS[15:0] | |
| R2 | DB3[7:0] | DB2[7:0] | DB1[7:0] | DB0[7:0] |
| R3 | DB7[7:0] | DB6[7:0] | DB5[7:0] | DB4[7:0] |
| ... | ... | ... | ... | ... |
| Rn | DBm[7:0] | DBm-1[7:0] | DBm-2[7:0] | DBm-3[7:0] |

**R0 Bit 31 ESI: Error State Indicator**

0: Transmitting node is error active
1: Transmitting node is error passive

**R0 Bit 30 XTD: Extended Identifier**

Signals to the Host whether the received frame has a standard or extended identifier.

0: 11-bit standard identifier
1: 29-bit extended identifier

**R0 Bit 29 RTR: Remote Transmission Request**

Signals to the Host whether the received frame is a data frame or a remote frame.

0:Received frame is a data frame

1:Received frame is a remote frame

**NOTE**

There are no remote frames in CAN FD format. In case a CAN FD frame (FDF = '1'), the dominant RRS (Remote Request Substitution) bit replaces bit RTR (Remote Transmission Request).

ISO CANFD ISO 11898-1:2015

**R0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

**R1 Bit 31 ANMF: Accepted Non-matching Frame**

Acceptance of non-matching frames may be enabled via MCANnGFC.ANFS and MCANnGFC.ANFE.

0:Received frame matching filter index FIDX

1:Received frame did not match any Rx filter element

**R1 Bits 30:24 FIDX[6:0]:Filter Index**

0 to 127:Index of matching Rx acceptance filter element (invalid if ANMF = '1').
Range is 0 to MCANnSIDFC.LSS - 1 resp. MCANnXIDFC.LSE - 1.

**R1 Bit 21 FDF: FD Format**

0:Standard frame format

1:CAN FD frame format (new DLC-coding and CRC)

**R1 Bit 20 BRS: Bit Rate Switch**

0:Frame received without bit rate switching

1:Frame received with bit rate switching

**R1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: received frame has 0 to 8 data bytes

9 to 15: CAN: received frame has 8 data bytes

9 to 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes

**R1 Bits 15:0 RXTS[15:0]:Rx Timestamp**

Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler MCANnTSCC.TCP.

ISO CANFD ISO 11898-1:2015

**R2 Bits 31:24 DB3[7:0]: Data Byte 3**

**R2 Bits 23:16 DB2[7:0]: Data Byte 2**

**R2 Bits 15:8 DB1[7:0]: Data Byte 1**

**R2 Bits 7:0 DB0[7:0]: Data Byte 0**

**R3 Bits 31:24 DB7[7:0]: Data Byte 7**

**R3 Bits 23:16 DB6[7:0]: Data Byte 6**

**R3 Bits 15:8 DB5[7:0]: Data Byte 5**

**R3 Bits 7:0 DB4[7:0]: Data Byte 4**

**...**

**Rn Bits 31:24 DBm[7:0]: Data Byte m**

**Rn Bits 23:16 DBm-1[7:0]:Data Byte m-1**

**Rn Bits 15:8 DBm-2[7:0]:Data Byte m-2**

**Rn Bits 7:0 DBm-3[7:0]:Data Byte m-3**

**NOTE**

Depending on the configuration of the element size (MCANnRXESC), between two and sixteen 32-bit words (Rn = 3 to 17) are used for storage of a CAN message's data field.

ISO CANFD ISO 11898-1:2015

### (3) Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration MCANnTXBC.TFQS and MCANnTXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register MCANnTXESC.

**Table 19.61 Tx Buffer Element**

| | 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|---|
| T0 | ESI, XTD, RTR | ID[28:0] | | |
| T1 | MM[7:0] | EFC, res, FDF, BRS, DLC[3:0] | res | |
| T2 | DB3[7:0] | DB2[7:0] | DB1[7:0] | DB0[7:0] |
| T3 | DB7[7:0] | DB6[7:0] | DB5[7:0] | DB4[7:0] |
| ... | ... | ... | ... | ... |
| Tn | DBm[7:0] | DBm-1[7:0] | DBm-2[7:0] | DBm-3[7:0] |

**T0 Bit 31 ESI: Error State Indicator**

0: ESI bit in CAN FD format depends only on error passive flag

1: ESI bit in CAN FD format transmitted recessive

**NOTE**

The ESI bit of the transmit buffer is or'ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive

**T0 Bit 30 XTD: Extended Identifier**

0: 11-bit standard identifier

1: 29-bit extended identifier

**T0 Bit 29 RTR: Remote Transmission Request**

0: Transmit data frame

1: Transmit remote frame

ISO CANFD ISO 11898-1:20

**NOTE**

When RTR = 1, the M_CAN transmits a remote frame according to ISO11898-1:2015, even if MCANnCCCR.FDOE enables the transmission in CAN FD format.

**T0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

**T1 Bits 31:24 MM[7:0]: Message Marker**

Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.

**T1 Bit 23 EFC: Event FIFO Control**

0: Don't store Tx events

1: Store Tx events

**T1 Bit 21 FDF: FD Format**

0: Frame transmitted in Classic CAN format

1: Frame transmitted in CAN FD format

**T1 Bit 20 BRS: Bit Rat Switching**

0: CAN FD frames transmitted without bit rate switching

1: CAN FD frames transmitted with bit rate switching

**NOTE**

Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled MCANnCCCR.FDOE = '1'. Bit BRS is only evaluated when in addition CCCR.BRSE = '1'

**T1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: transmit frame has 0-8 data bytes

9 to 15: CAN: transmit frame has 8 data bytes

9 to 15: CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes

ISO CANFD ISO 11898-1:2015

**T2 Bits 31:24 DB3[7:0]:Data Byte 3**

**T2 Bits 23:16 DB2[7:0]: Data Byte 2**

**T2 Bits 15:8 DB1[7:0]: Data Byte 1**

**T2 Bits 7:0 DB0[7:0]: Data Byte 0**

**T3 Bits 31:24 DB7[7:0]: Data Byte 7**

**T3 Bits 23:16 DB6[7:0]: Data Byte 6**

**T3 Bits 15:8 DB5[7:0]: Data Byte 5**

**T3 Bits 7:0 DB4[7:0]: Data Byte 4**

**...**

**Tn Bits 31:24 DBm[7:0]: Data Byte m**

**Tn Bits 23:16 DBm-1[7:0]:Data Byte m-1**

**Tn Bits 15:8 DBm-2[7:0]:Data Byte m-2**

**Tn Bits 7:0 DBm-3[7:0]:Data Byte m-3**

**NOTE**

Depending on the configuration of the element size (MCANnTXESC), between two and sixteen 32-bit words (Tn = 3 to 17) are used for storage of a CAN message's data field.

ISO CANFD ISO 11898-1:2015

### (4) Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register MCANnTXEFS

**Table 19.62 Tx Event FIFO Element**

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E0 | ESI | XTD | RTR | ID[28:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E1 | MM[7:0] | | | | | | | | ET[1:0] | | FDF | BRS | DLC[3:0] | | | | TXTS[15:0] | | | | | | | | | | | | | | | |

**E0 Bit 31 ESI: Error State Indicator**

0: Transmitting node is error active

1: Transmitting node is error passive

**E0 Bit 30 XTD: Extended Identifier**

0: 11-bit standard identifier

1: 29-bit extended identifier

**E0 Bit 29 RTR: Remote Transmission Request**

0: Data frame transmitted

1: Remote frame transmitted

**E0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

**E1 Bits 31:24 MM[7:0]: Message Marker**

Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.

**E1 Bit 23:22 ET[1:0]: Event Type**

00: Reserved

01: Tx event

10: Transmission in spite of cancellation (always set for transmissions in DAR mode)

11: Reserved

**E1 Bit 21 FDF: FD Format**

0: Standard frame format

1: CAN FD frame format (new DLC-coding and CRC)

**E1 Bit 20 BRS: Bit Rate Switch**

0: Frame transmitted without bit rate switching

1: Frame transmitted with bit rate switching

ISO CANFD ISO 11898-1:2015

**E1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: frame with 0-8 data bytes transmitted

9 to 15: CAN: frame with 8 data bytes transmitted

9 to 15: CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted

**E1 Bits 15:0 TXTS[15:0]:Tx Timestamp**

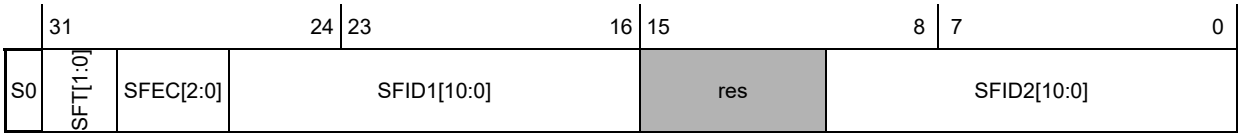
Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler MCANnTSCC.TCP.

ISO CANFD ISO 11898-1:2015

### (5) Standard Message ID Filter Element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address MCANnSIDFC.FLSSA plus the index of the filter element (0 to 127).

**Table 19.63 Standard Message ID Filter Element**




**Bits 31:30 SFT[1:0]: Standard Filter Type**

00: Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1)

01: Dual ID filter for SFID1 or SFID2

10: Classic filter: SFID1 = filter, SFID2 = mask

11: Filter element disabled

**NOTE**

With SFT = “11” the filter element is disabled and the acceptance filtering continues (same behaviour as with SFEC = “000”)

**Bit 29:27 SFEC[2:0]:Standard Filter Element Configuration**

All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag MCANnIR.HPM and, if enabled, an interrupt is generated. In this case register MCANnHPMS is updated with the status of the priority match.

000: Disable filter element

001: Store in Rx FIFO 0 if filter matches

010: Store in Rx FIFO 1 if filter matches

011: Reject ID if filter matches

100: Set priority if filter matches

101: Set priority and store in FIFO 0 if filter matches

110: Set priority and store in FIFO 1 if filter matches

111: Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored

**Bits 26:16 SFID1[10:0]:Standard Filter ID 1**

First ID of standard ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.

ISO CANFD ISO 11898-1:2015

**Bits 10:0 SFID2[10:0]:Standard Filter ID 2**

This bit field has a different meaning depending on the configuration of SFEC:

1) SFEC = “001” to “110”Second ID of standard ID filter element

2) SFEC = “111”Filter for Rx Buffers or for debug messages

**SFID2[10:9]**: These bits decide whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.
00: Store message into an Rx Buffer
01: Debug Message A
10: Debug Message B
11: Debug Message C

**SFID2[8:6]**: These bits are used to control the filter event pins m_can_fe[2:0] at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_can_hclk (CLK_HSB) period in case the filter matches.

**SFID2[5:0]:** These bits define the offset to the Rx Buffer Start Address MCANnRXBC.RBSA for storage of a matching message.

ISO CANFD ISO 11898-1:2015

### (6) Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address MCANnXIDFC.FLESA plus two times the index of the filter element (0 to 63).

**Table 19.64 Extended Message ID Filter Element**

| | 31 | 24 | 23 | 16 | 15 | 8 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|
| F0 | EFEC[2:0] | | EFID1[28:0] | | | | | |
| F1 | EFT[1:0] | res | EFID2[28:0] | | | | | |

#### F0 Bit 31:29 EFEC[2:0]:Extended Filter Element Configuration

All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = “100”, “101”, or “110” a match sets interrupt flag MCANnIR.HPM and, if enabled, an interrupt is generated. In this case register MCANnHPMS is updated with the status of the priority match.

000: Disable filter element

001: Store in Rx FIFO 0 if filter matches

010: Store in Rx FIFO 1 if filter matches

011: Reject ID if filter matches

100: Set priority if filter matches

101: Set priority and store in FIFO 0 if filter matches

110: Set priority and store in FIFO 1 if filter matches

111: Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored

#### F0 Bits 28:0 EFID1[28:0]:Extended Filter ID 1

First ID of extended ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only MCANnXIDAM masking mechanism (see **Section (e), Extended Message ID Filtering**) is used.

#### F1 Bits 31:30 EFT[1:0]: Extended Filter Type

00: Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1)

01: Dual ID filter for EFID1 or EFID2

10: Classic filter: EFID1 = filter, EFID2 = mask

11: Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), MCANnXIDAM mask not applied

ISO CANFD ISO 11898-1:2015

**F1 Bits 28:0 EFID2[28:0]:Extended Filter ID 2**

This bit field has a different meaning depending on the configuration of EFEC:

1) EFEC: “001” to ”110”Second ID of extended ID filter element

2) EFEC: “111”Filter for Rx Buffers or for debug messages

**EFID2[10:9]**: These bits decide whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.
00: Store message into an Rx Buffer
01: Debug Message A
10: Debug Message B
11: Debug Message C

**EFID2[8:6]**: These bits are used to control the filter event pins m_can_fe[2:0] at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_can_hclk (CLK_HSB) period in case the filter matches.

**EFID2[5:0]:** These bits define the offset to the Rx Buffer Start Address MCANnRXBC.RBSA for storage of a matching message.

ISO CANFD ISO 11898-1:2015

## 19.5.3 Functional Description

### 19.5.3.1 Operating Modes

#### (1) Software Initialization

Software initialization is started by setting bit MCANnCCCR.INIT, either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While MCANnCCCR.INIT is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output m_can_tx is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting MCANnCCCR.INIT does not change any configuration register. Resetting MCANnCCCR.INIT finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (≡ Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits MCANnCCCR.INIT and MCANnCCCR.CCE are set (protected write).

MCANnCCCR.CCE can only be set/reset while MCANnCCCR.INIT = '1'. MCANnCCCR.CCE is automatically reset when MCANnCCCR.INIT is reset.

The following registers are reset when MCANnCCCR.CCE is set

- MCANnHPMS - High Priority Message Status
- MCANnRXF0S - Rx FIFO 0 Status
- MCANnRXF1S - Rx FIFO 1 Status
- MCANnTXFQS - Tx FIFO/Queue Status
- MCANnTXBRP - Tx Buffer Request Pending
- MCANnTXBTO - Tx Buffer Transmission Occurred
- MCANnTXBCF - Tx Buffer Cancellation Finished
- MCANnTXEFS - Tx Event FIFO Status

The Timeout Counter value MCANnTOCV.TOC is preset to the value configured by MCANnTOCC.TOP when MCANnCCCR.CCE is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while MCANnCCCR.CCE = '1'.

The following registers are only writable while MCANnCCCR.CCE = '0'

- MCANnTXBAR - Tx Buffer Add Request
- MCANnTXBCR - Tx Buffer Cancellation Request

MCANnCCCR.TEST and MCANnCCCR.MON can only be set by the Host while MCANnCCCR.INIT = '1' and MCANnCCCR.CCE = '1'. Both bits may be reset at any time. MCANnCCCR.DAR can only be set/reset while MCANnCCCR.INIT = '1' and MCANnCCCR.CCE = '1'.

ISO CANFD ISO 11898-1:2015

### (2) Normal Operation

Once the M_TTCAN is initialized and MCANnCCCR.INIT is reset to *zero*, the M_CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

### (3) CAN FD Operation

There are two variants in the CAN FD frame transmission, first the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = recessive signifies a CAN FD frame, FDF = dominant signifies a Classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. The coding of res = recessive is reserved for future expansion of the protocol. In case the M_CAN receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting bit PSR.PXE. When Protocol Exception Handling is enabled (MCANnCCCR.PXHD = '0'), this causes the operation state to change from Receiver (MCANnPSR.ACT = "10") to Integrating (MCANnPSR.ACT = "00") at the next sample point. In case Protocol Exception Handling is disabled (MCANnCCCR.PXHD = '1'), the M_CAN will treat a recessive res bit as an form error and will respond with an error frame.

ISO CANFD ISO 11898-1:2015

CAN FD operation is enabled by programming MCANnCCCR.FDOE. In case MCANnCCCR.FDOE = '1', transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via bit FDF in the respective Tx Buffer element. With MCANnCCCR.FDOE = '0', received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx Buffer element is set. MCANnCCCR.FDOE and MCANnCCCR.BRSE can only be changed while MCANnCCCR.INIT and MCANnCCCR.CCE are both set.

With MCANnCCCR.FDOE = '0', the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With MCANnCCCR.FDOE = '1' and MCANnCCCR.BRSE = '0', only bit FDF of a Tx Buffer element is evaluated. With MCANnCCCR.FDOE = '1' and MCANnCCCR.BRSE = '1', transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD

operation.

- Wake-up messages in CAN Partial Networking have to be transmitted in Classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to **Table 19.65, Coding of DLC in CAN FD** below.

**Table 19.65 Coding of DLC in CAN FD**

| DLC | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| Number of Data Bytes | 12 | 16 | 20 | 24 | 32 | 48 | 64 |

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Nominal Bit Timing & Prescaler Register MCANnNBTP. In the following CAN FD data phase, the data phase bit timing is used as defined by the Data Bit Timing & Prescaler Register MCANnDBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (m_can_cclk: CLKP_H2). Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long and CAN FD with bit rate switching , the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

ISO CANFD ISO 11898-1:2015

### (4) Transmitter Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin m_can_tx the protocol controller receives the transmitted data from its local CAN transceiver via pin m_can_rx. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transceiver loop delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

#### (a) Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the ISO11898-1:2015. It is enabled by setting bit MCANnDBTP.TDC.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the CAN FD transmit output m_can_tx through the transceiver to the receive input m_can_rx plus the transmitter delay compensation offset as configured by MCANnTDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq.

MCANnPSR.TDCV shows the actual transmitter delay compensation value. MCANnPSR.TDCV is cleared when MCANnCCCR.INIT is set and is updated at each transmission of an FD frame while MCANnDBTP.TDC is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the M_CAN:

- The sum of the measured delay from m_can_tx to m_can_rx and the configured transmitter delay compensation offset MCANnTDCR.TDCO has to be less than 6 bit times in the data phase.
- The sum of the measured delay from m_can_tx to m_can_rx and the configured transmitter delay compensation offset MCANnTDCR.TDCO has to be less or equal 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs

(b) Transmitter Delay Compensation Measurement

If transmitter delay compensation is enabled by programming MCANnDBTP.TDC = '1', the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input m_can_rx of the transmitter. The resolution of this measurement is one mtq

ISO CANFD ISO 11898-1:2015

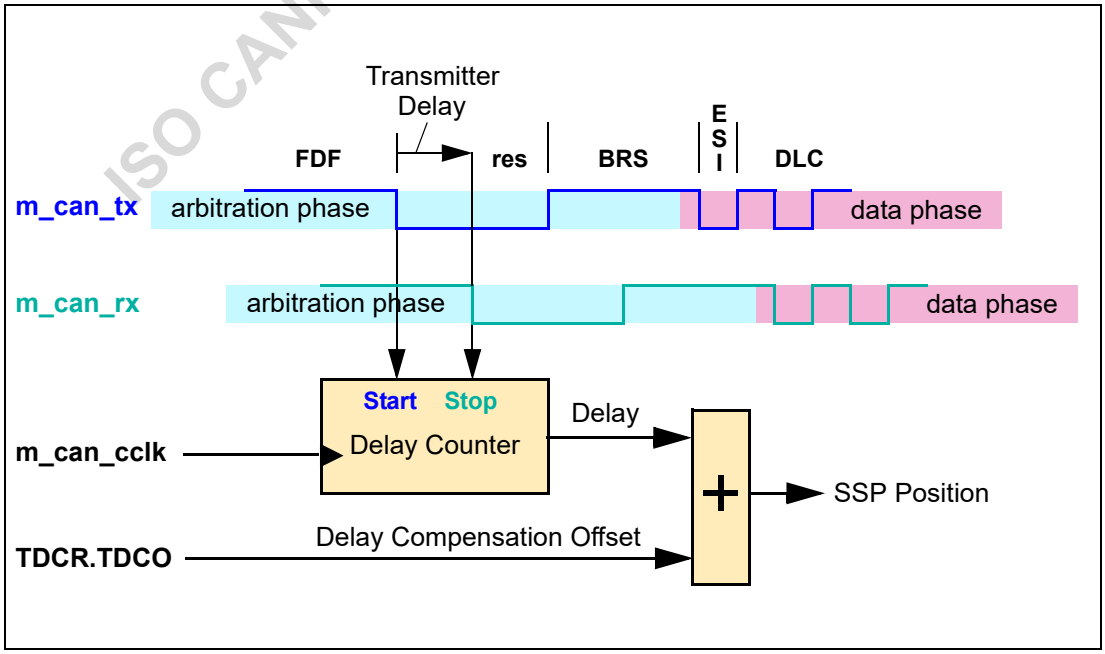



**Figure 19.5 Transmitter delay measurement**

To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a to early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming MCANnTDCR.TDCF.

This defines a minimum value for the SSP position. Dominant edges on m_can_rx, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least MCANnTDCR.TDCF AND m_can_rx is low.

### (5) Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters(MCANnECR.REC, MCANnECR.TEC) are frozen while Error Logging (MCANnECR.CEL) is active. The Host can set the M_CAN into Restricted Operation mode by setting bit MCANnCCCR.ASM. The bit can only be set by the Host when both MCANnCCCR.CCE and MCANnCCCR.INIT are set to '1'. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset MCANnCCCR.ASM.

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

If the M_CAN is connected to a Clock Calibration on CAN unit, MCANnCCCR.ASM is controlled by input m_can_cok. In case m_can_cok switches to '0', bit MCANnCCCR.ASM is set. When m_can_cok switches back to '1', bit MCANnCCCR.ASM returns to the previously written value. When there is no Clock Calibration on CAN unit connected input m_can_cok is hardwired to '1'.

**NOTE**

The Restricted Operation Mode must not be combined with the Loop Back Mode (internal or external).

ISO CAN FD ISO 11898-1:2015

### (6) Bus Monitoring Mode

The M_CAN is set in Bus Monitoring Mode by programming MCANnCCCR.MON to one. In Bus Monitoring Mode (see ISO11898-1:2015, 10.12 Bus monitoring), the M_TTCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the M_TTCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_TTCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode register MCANnTXBRP is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. **Figure 19.6** shows the connection of signals m_can_tx and m_can_rx to the M_CAN in Bus Monitoring Mode.

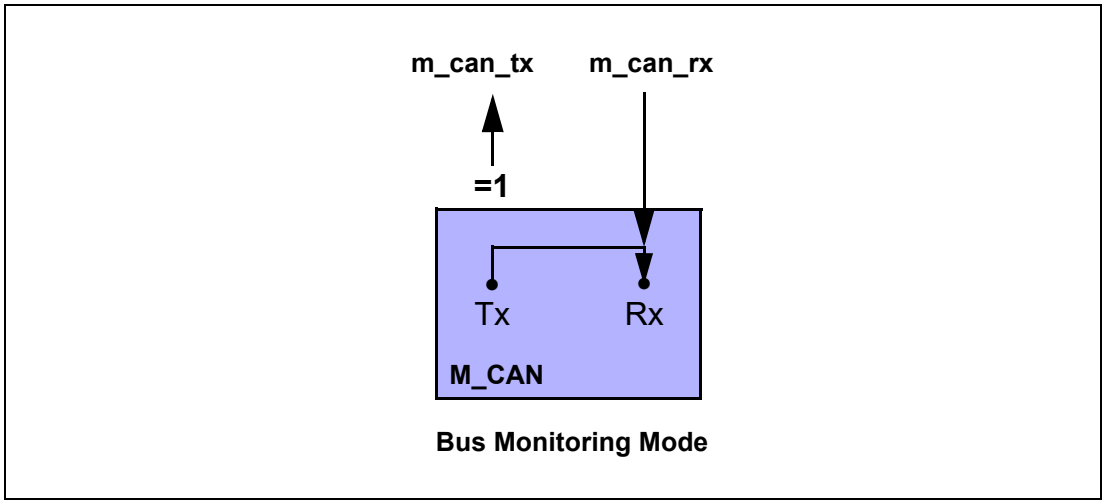



Figure 19.6 Pin Control in Bus Monitoring Mode

ISO CANFD ISO 11898-1:2015

### (7) Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898-1:2015, 6.3.3 Recovery Management), the M_TTCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via MCANnCCCR.DAR.

#### (a) Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit MCANnTXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  Corresponding Tx Buffer Transmission Occurred bit MCANnTXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit MCANnTXBCF.CFx not set
- Successful transmission in spite of cancellation:
  Corresponding Tx Buffer Transmission Occurred bit MCANnTXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit MCANnTXBCF.CFx set
- Arbitration lost or frame transmission disturbed:
  Corresponding Tx Buffer Transmission Occurred bit MCANnTXBTO.TOx not set
  Corresponding Tx Buffer Cancellation Finished bit MCANnTXBCF.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = "10" (transmission in spite of cancellation).

### (8) Power Down (Sleep Mode)

The M_TTCAN can be set into power down mode by using CC Control Register MCANnCCCR.CSR. When all pending transmission requests have completed, the M_TTCAN waits until bus idle state is detected. Then the M_TTCAN sets then MCANnCCCR.INIT to one to prevent any further CAN transfers. Now the M_TTCAN acknowledges that it is ready for power down by setting MCANnCCCR.CSA to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to MCANnCCCR.INIT will have no effect. Now the module clock inputs m_can_hclk (CLK_HSB) and m_can_cclk (CLKP_H2) may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting CC Control Register flag MCANnCCCR.CSR. The M_CAN will acknowledge this by resetting MCANnCCCR.CSA. Afterwards, the application can restart CAN communication by resetting bit MCANnCCCR.INIT.

### (9) Test Modes

To enable write access to register MCANnTEST (see **(4) MCANnTEST — Test Register**), bit MCANnCCCR.TEST has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin m_can_tx by programming MCANnTEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_CAN's bit timing and it can drive constant dominant or recessive values. The actual value at pin m_can_rx can be read from MCANnTEST.RX. Both functions can be used to check the CAN bus' physical layer.

ISO CAN FD ISO 11898-1:2015

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to MCANnTEST.TX until the new configuration is visible at output pin m_can_tx. This applies also when reading input pin m_can_rx via MCANnTEST.RX.

**NOTE**

Test modes should be used for production tests or self test only. The software control for pin m_can_tx interferes with all CAN protocol functions. It is not recommended to use test modes for application.

(a) External Loop Back Mode

The M_CAN can be set in External Loop Back Mode by programming MCANnTEST.LBCK to one. In Loop Back Mode, the M_CAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx Buffer or an Rx FIFO. **Figure 19.7, Pin Control in Loop Back Modes** shows the connection of signals m_can_tx and m_can_rx to the M_CAN in External Loop Back Mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the M_CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the m_can_rx input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the m_can_tx pin.

(b) Internal Loop Back Mode

Internal Loop Back Mode is entered by programming bits MCANnTEST.LBCK and MCANnCCCR.MON to one. This mode can be used for a “Hot Selftest”, meaning the M_TTCAN can be tested without affecting a running CAN system connected to the pins m_can_tx and m_can_rx. In this mode pin m_can_rx is disconnected from the M_CAN and pin m_can_tx is held recessive. **Figure 19.7, Pin Control in Loop Back Modes** shows the connection of m_can_tx and m_can_rx to the M_CAN in case of Internal Loop Back Mode.

ISO CAN FD ISO 11898-1:2015

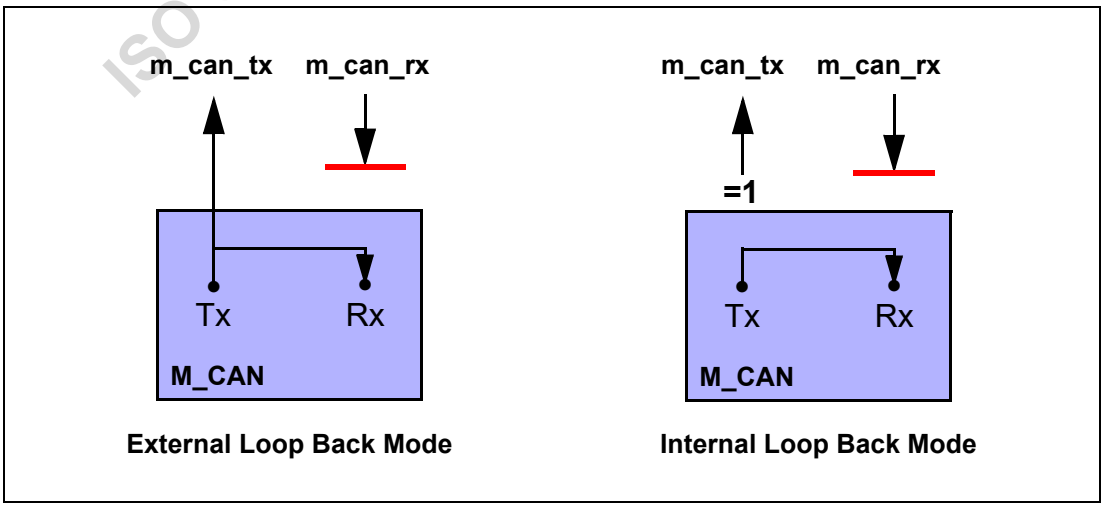



Figure 19.7 Pin Control in Loop Back Modes

### 19.5.3.2 Timestamp Generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler MCANnTSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via MCANnTSCV.TSC. A write access to register MCANnTSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag MCANnIR.TSW is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer / Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit MCANnTSCC.TSS an external 16-bit timestamp can be used.

### 19.5.3.3 Timeout Counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_CAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by MCANnTSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via register MCANnTOCC. The actual counter value can be read from MCANnTOCV.TOC. The Timeout Counter can only be started while MCANnCCCR.INIT = '0'. It is stopped when MCANnCCCR.INIT = '1', e.g. when the M_CAN enters Bus_Off state.

The operation mode is selected by MCANnTOCC.TOS. When operating in Continuous Mode, the counter starts when MCANnCCCR.INIT is reset. A write to MCANnTOCV presets the counter to the value configured by MCANnTOCC.TOP and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by MCANnTOCC.TOP. Down-counting is started when the first FIFO element is stored. Writing to MCANnTOCV has no effect.

When the counter reaches zero, interrupt flag MCANnIR.TOO is set. In Continuous Mode, the counter is immediately restarted at MCANnTOCC.TOP.

**NOTE**

The clock signal for the Timeout Counter is derived from the CAN Core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the bit rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

ISO CANFD ISO 11898-1:2015

### 19.5.3.4 Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

#### (1) Acceptance Filtering

The M_CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration MCANnGFC
- Standard ID Filter Configuration MCANnSIDFC
- Extended ID Filter Configuration MCANnXIDFC
- Extended ID AND Mask MCANnXIDAM

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag MCANnIR.HPM
- Set High Priority Message interrupt flag MCANnIR.HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

ISO CAN FD ISO 11898-1:2015

**Rx Buffer**

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see MCANnPSR.LEC respectively MCANnPSR.DLEC.

**Rx FIFO**

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see MCANnPSR.LEC respectively MCANnPSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in **(b) Rx FIFO Overwrite Mode** have to be considered.

**NOTE**

When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

(a) Range Filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

EFT: “00”: The Message ID of received frames is ANDed with the Extended ID AND Mask (MCANnXIDAM) before the range filter is applied

EFT: “11”: The Extended ID AND Mask (MCANnXIDAM) is not used for range filtering

(b) Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

(c) Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

ISO CAN FD ISO 11898-1:2015

(d) Standard Message ID Filtering

**Figure 19.8, Standard Message ID Filter Path** below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in **(5) Standard Message ID Filter Element**.

Controlled by the Global Filter Configuration MCANnGFC and the Standard ID Filter Configuration MCANnSIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

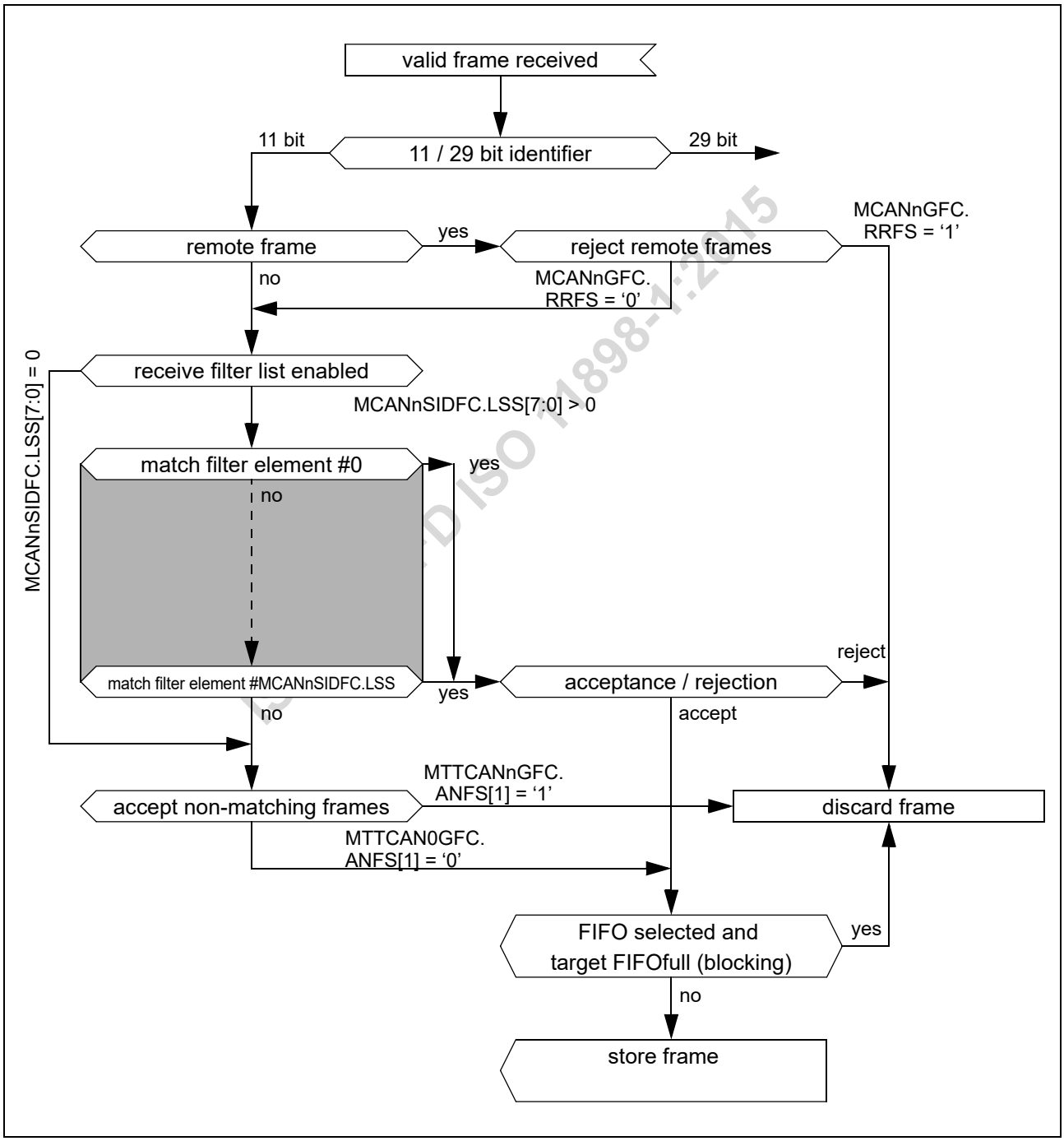



**Figure 19.8 Standard Message ID Filter Path**

(e) Extended Message ID Filtering

**Figure 19.9, Extended Message ID Filter Path** below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in **(6) Extended Message ID Filter Element**.

Controlled by the Global Filter Configuration MCANnGFC and the Extended ID Filter Configuration MCANnXIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask MCANnXIDAM is ANDed with the received identifier before the filter list is executed.

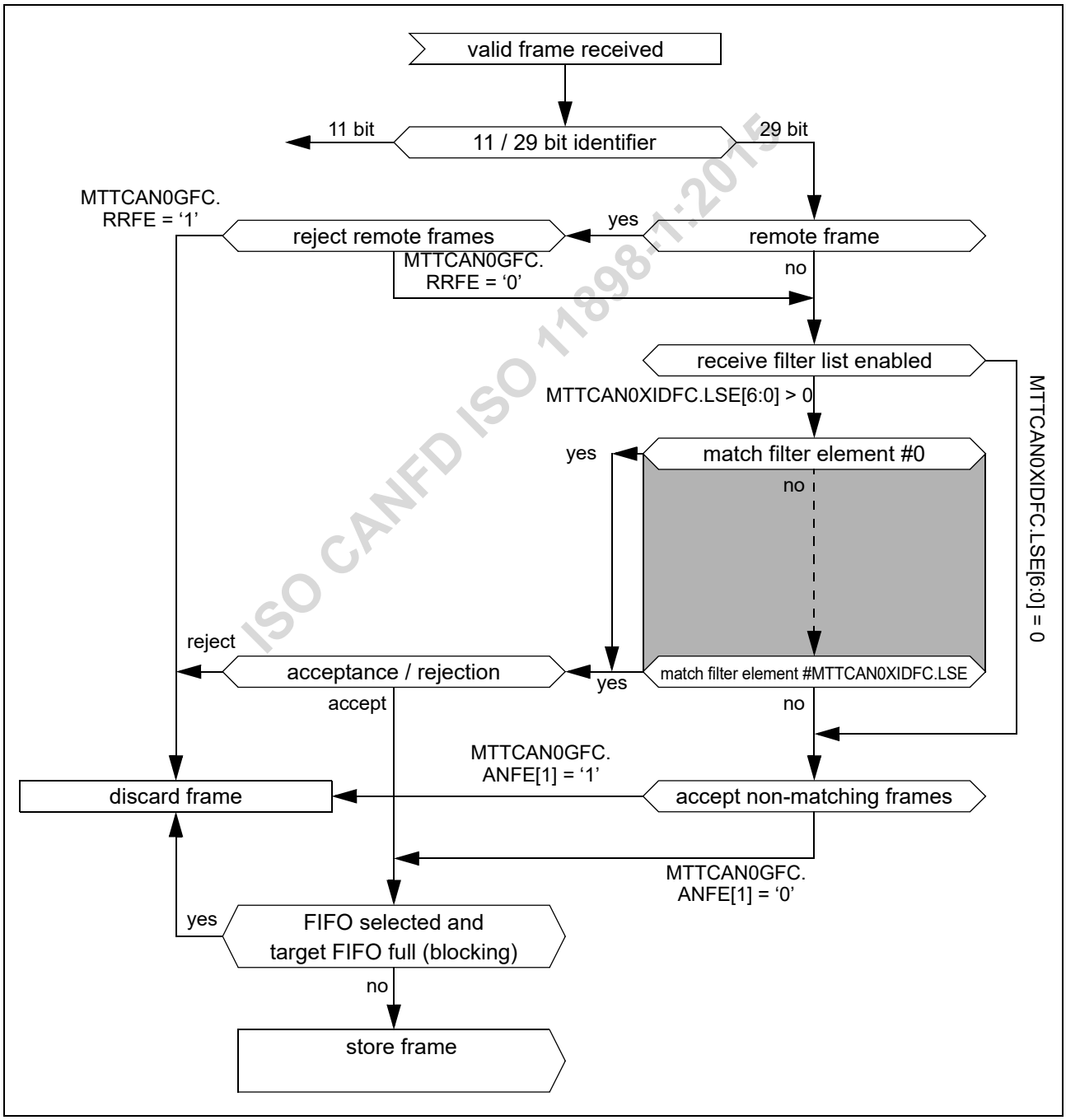



**Figure 19.9 Extended Message ID Filter Path**

### (2) Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers MCANnRXF0C and MCANnRXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see **(1) Acceptance Filtering**. The Rx FIFO element is described in **(2) Rx Buffer and FIFO Element**.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag MCANnIR.RFnW is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS.FnF. In addition interrupt flag MCANnIR.RFnF is set.

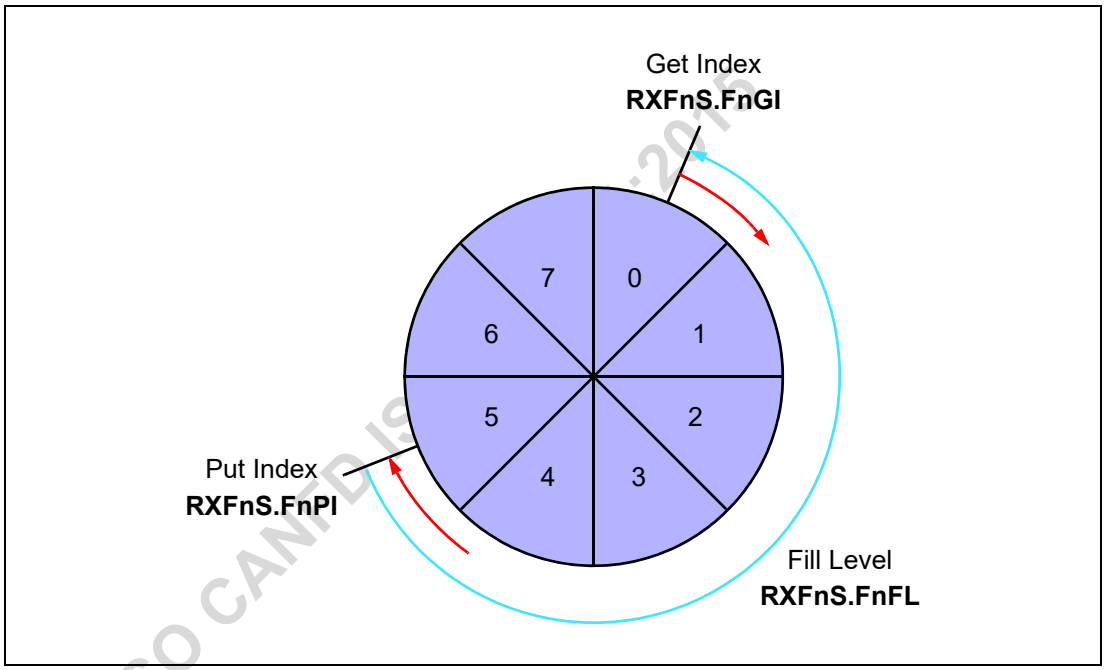



Figure 19.10 Rx FIFO Status

When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI • FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

Table 19.66 Rx Buffer / FIFO Element Size

| MCANnRXESC.RBDS[2:0]<br>MCANnRXESC.FnDS[2:0] | Data Field<br>[bytes] | FIFO Element Size<br>[RAM words] |
|---|---|---|
| 000 | 8 | 4 |
| 001 | 12 | 5 |
| 010 | 16 | 6 |
| 011 | 20 | 7 |
| 100 | 24 | 8 |
| 101 | 32 | 10 |
| 110 | 48 | 14 |
| 111 | 64 | 18 |

(a) Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = '0'. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS.FnF = '1'. In addition interrupt flag MCANnIR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = '1'. In addition interrupt flag MCANnIR.RFnL is set.

(b) Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = '1'.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signalled by RXFnS.FnF = '1', the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. **Figure 19.11, Rx FIFO Overflow Handling** shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

ISO CANFD ISO 11898-1:2015

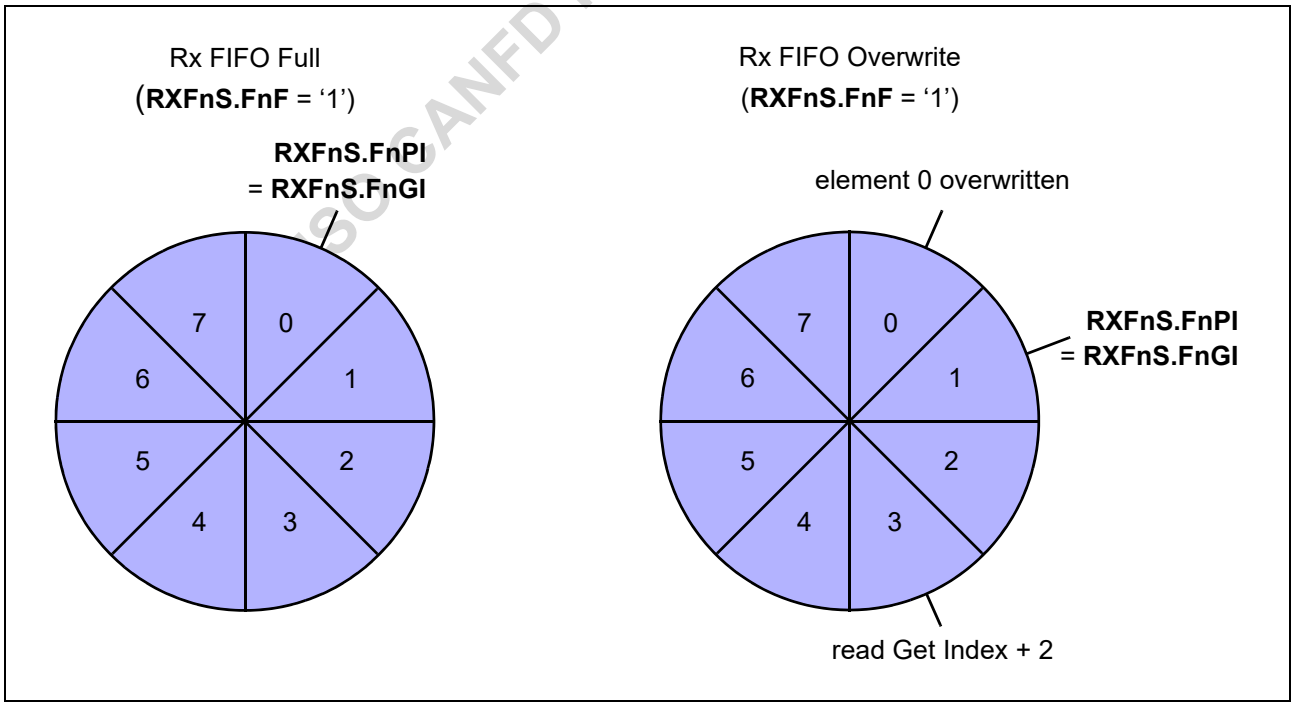



Figure 19.11 Rx FIFO Overflow Handling

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = '0').

### (3) Dedicated Rx Buffers

The M_CAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via MCANnRXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = “111” and SFID2 / EFID2[10:9] = “00” has to be configured (see **(5) Standard Message ID Filter Element** and **(6) Extended Message ID Filter Element**).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag MCANnIR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

**Table 19.67 Example Filter Configuration for Rx Buffers**

| Filter Element | SFID1[10:0] EFID1[28:0] | SFID2[10:9] EFID2[10:9] | SFID2[5:0] EFID2[5:0] |
|---|---|---|---|
| 0 | ID message 1 | 00 | 00 0000 |
| 1 | ID message 2 | 00 | 00 0001 |
| 2 | ID message 3 | 00 | 00 0010 |

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register MCANnNDAT1, 2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a ‘1’ to the respective bit position.

While an Rx Buffer’s New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

#### (a) Rx Buffer Handling

- Reset interrupt flag MCANnIR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

### (4) Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (e.g. #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (see M_CAN User’s Manual section 2.4.2).

Advantage: Fixed start address for the DMA transfers (relative to MCANnRXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = “111” have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output m_can_dma_req is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M_CAN while m_can_dma_req is activated. The behaviour is similar to that of an Rx Buffers with its New Data flag set.

ISO CANFD ISO 11898-1:2015

After the DMA has completed the DMA unit sets m_ttcan_dma_ack. This resets m_can_dma_req. Now the M_TTCAN is prepared to receive the next set of debug messages.

### (a) Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to “111”. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning (see Section 2.4.5 and Section 2.4.6). While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor MCANnIR.DRX are set. The reception of debug messages can be monitored via MCANnRXF1S.DMS

**Table 19.68 Example Filter Configuration for Debug Message**

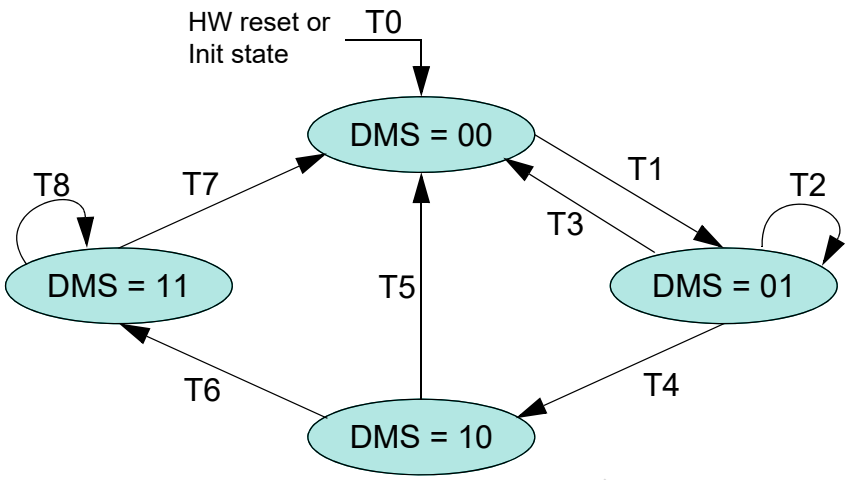
| Filter Element | SFID1[10:0] EFID1[28:0] | SFID2[10:9] EFID2[10:9] | SFID2[5:0] EFID2[5:0] |
|---|---|---|---|
| 0 | ID debug message A | 01 | 11 1101 |
| 1 | ID debug message B | 10 | 11 1110 |
| 2 | ID debug message C | 11 | 11 1111 |

### (b) Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signalled via RXF1S.DMS

ISO CANFD ISO 11898-1:201




T0: reset m_can_dma_req output, enable reception of debug messages A, B, and C
T1: reception of debug message A
T2: reception of debug message A
T3: reception of debug message C
T4: reception of debug message B
T5: reception of debug messages A, B
T6: reception of debug message C
T7: DMA transfer completed
T8: reception of debug message A,B,C (message rejected)

**Figure 19.12 Debug Message Handling State Machine**

ISO CANFD ISO 11898-1:2015

### 19.5.3.5 Tx Handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The CAN mode for transmission (Classic CAN or CAN FD) can be configured separately for each Tx Buffer element. The Tx Buffer element is described in **(3) Tx Buffer Element**. **Table 19.69** below describes the possible configurations for frame transmission

**Table 19.69 Possible Configurations for Frame Transmission**

| CCCR | | Tx Buffer Element | | |
|---|---|---|---|---|
| BRSE | FDOE | FDF | BRS | Frame Transmission |
| ignored | 0 | ignored | ignored | Classic CAN |
| 0 | 1 | 0 | ignored | Classic CAN |
| 0 | 1 | 1 | ignored | FD without bit rate switching |
| 1 | 1 | 0 | ignored | Classic CAN |
| 1 | 1 | 1 | 0 | FD without bit rate switching |
| 1 | 1 | 1 | 1 | FD with bit rate switching |

**NOTE**

AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register MCANnTXBRP is updated, or when a transmission has been started.

#### (1) Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit MCANnCCCR.TXP. If the bit is set, the M_CAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (MCANnCCCR.TXP = '0').

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

ISO CANFD ISO 11898-1:2015

### (2) Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via MCANnTXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (see **Table 19.70, Tx Buffer / FIFO / Queue Element Size**). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0 to 31) • Element Size to the Tx Buffer Start Address MCANnTXBC.TBSA.

**Table 19.70 Tx Buffer / FIFO / Queue Element Size**

| MCANnTXESC.TBDS[2:0] | Data Field [bytes] | Element Size [RAM words] |
|---|---|---|
| 000 | 8 | 4 |
| 001 | 12 | 5 |
| 010 | 16 | 6 |
| 011 | 20 | 7 |
| 100 | 24 | 8 |
| 101 | 32 | 10 |
| 110 | 48 | 14 |
| 111 | 64 | 18 |

ISO CANFD ISO 11898-1:2015

### (3) Tx FIFO

Tx FIFO operation is configured by programming MCANnTXBC.TFQM to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index MCANnTXFQS.TFGI. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_CAN calculates the Tx FIFO Free Level MCANnTXFQS.TFFL as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index MCANnTXFQS.TFQPI. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (MCANnTXFQS.TFQF = '1') is signalled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the MCANnTXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via MCANnTXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see **Table 19.70, Tx Buffer / FIFO / Queue Element Size**). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index MCANnTXFQS.TFQPI (0 to 31) • Element Size to the Tx Buffer Start Address MCANnTXBC.TBSA.

ISO CAN FD ISO 11898-1:2015

### (4) Tx Queue

Tx Queue operation is configured by programming MCANnTXBC.TFQM to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index MCANnTXFQS.TFQPI. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (MCANnTXFQS.TFQF = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register MCANnTXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see **Table 19.70, Tx Buffer / FIFO / Queue Element Size**). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index MCANnTXFQS.TFQPI (0 to 31) • Element Size to the Tx Buffer Start Address MCANnTXBC.TBSA.

### (5) Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by MCANnTXBC.NDTB. The number of Tx Buffers assigned to the Tx FIFO is configured by MCANnTXBC.TFQS. In case MCANnTXBC.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

ISO CANFD ISO 11898-1:2015

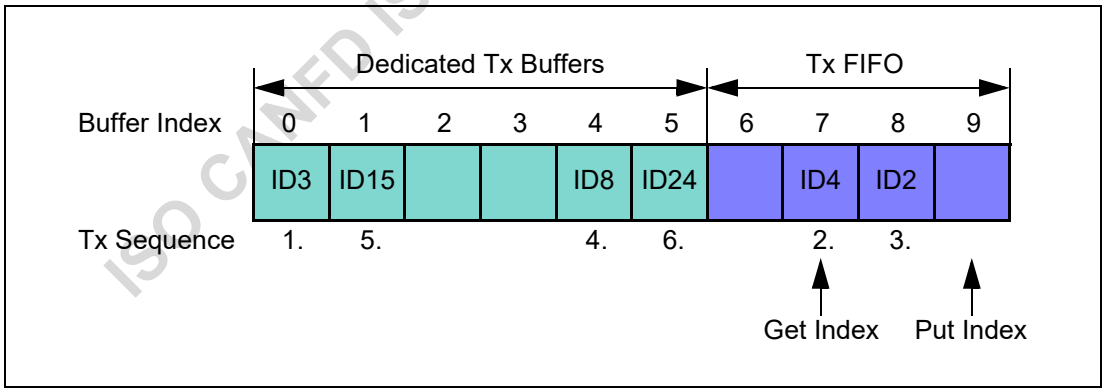



**Figure 19.13 Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO**

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS.TFGI)
- Buffer with lowest Message ID gets highest priority and is transmitted next

### (6) Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by MCANnTXBC.NDTB. The number of Tx Queue Buffers is configured by MCANnTXBC.TFQS. In case MCANnTXBC.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

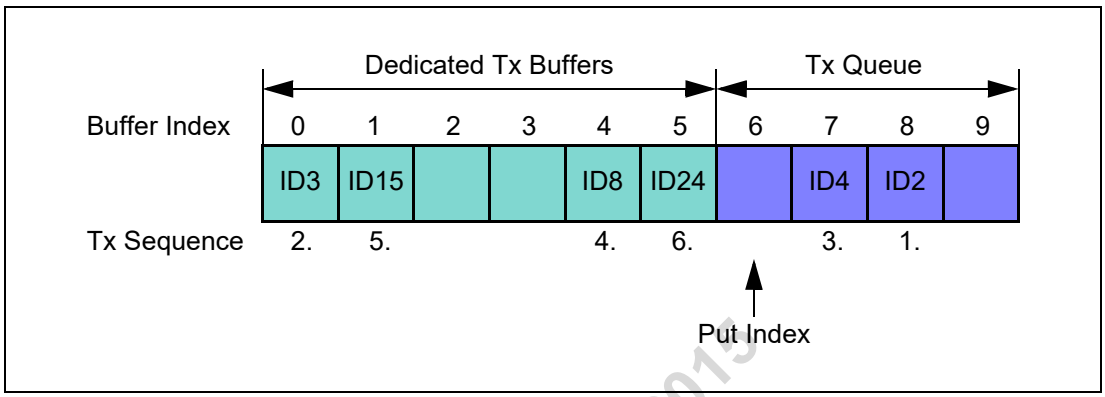



**Figure 19.14 Example of mixed Configuration Dedicated Tx Buffers / Tx Queue**

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

### (7) Transmit Cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (= number of Tx Buffer) of register MCANnTXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signalled by setting the corresponding bit of register MCANnTXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding MCANnTXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding MCANnTXBTO and MCANnTXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding MCANnTXBCF bit is set.

**NOTE**

In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

ISO CAN FD ISO 11898-1:2015

### (8) Tx Event Handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in **(4) Tx Event FIFO Element**.

The purpose of the Tx Event FIFO is to decouple handling transmit status information from transmit message handling i.e. a Tx Buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx Event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx Buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx Buffer before overwriting that Tx Buffer.

When a Tx Event FIFO full condition is signalled by MCANnIR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag MCANnIR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by MCANnTXEFC.EFWM, interrupt flag MCANnIR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index MCANnTXEFS.EFGI has to be added to the Tx Event FIFO start address MCANnTXEFC.EFSA.

ISO CANFD ISO 11898-1:2015

#### 19.5.3.6 FIFO Acknowledge Handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see **(28) MCANnRXF0A — Rx FIFO 0 Acknowledge**, **(32) MCANnRXF1A — Rx FIFO 1 Acknowledge**, and **(46) MCANnTXEFA — Tx Event FIFO Acknowledge**). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus *one* and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.

When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the M_CAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

**NOTE**

The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

ISO CANFD ISO 11898-1:2015

# 19.6 M_TTCAN

## 19.6.1 Overview

The M_TTCAN module is the new TTCAN Communication Controller IP-module. The M_TTCAN performs communication according to ISO 11898-1:2015 and according to ISO 11898-4 (Time-triggered communication on CAN). The M_TTCAN provides all features of time-triggered communication specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation. The CAN FD option can be used together with event-triggered and time-triggered CAN communication.

The message storage is intended to be a single-ported Message RAM outside of the module. It is connected to the M_TTCAN via the Generic Master Interface.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN Core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN Core as well as providing transmit status information. It implements all functions concerning the time schedule and the global system time.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_TTCAN's clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FM-PLL.

ISO CANFD ISO 11898-1:2015

#### 19.6.1.1 Features

- Conform with ISO11898-1:2015 and ISO 11898-4
- CAN FD with up to 64 data bytes supported
- TTCAN protocol level 1 and level 2 completely in hardware
- Event synchronized time-triggered communication supported
- CAN Error Logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signalling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Programmable loop-back test mode
- Maskable module interrupts
- 8/16/32 bit Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support
- Debug on CAN support

ISO CANFD ISO 11898-1:2015

### 19.6.1.2 Block Diagram

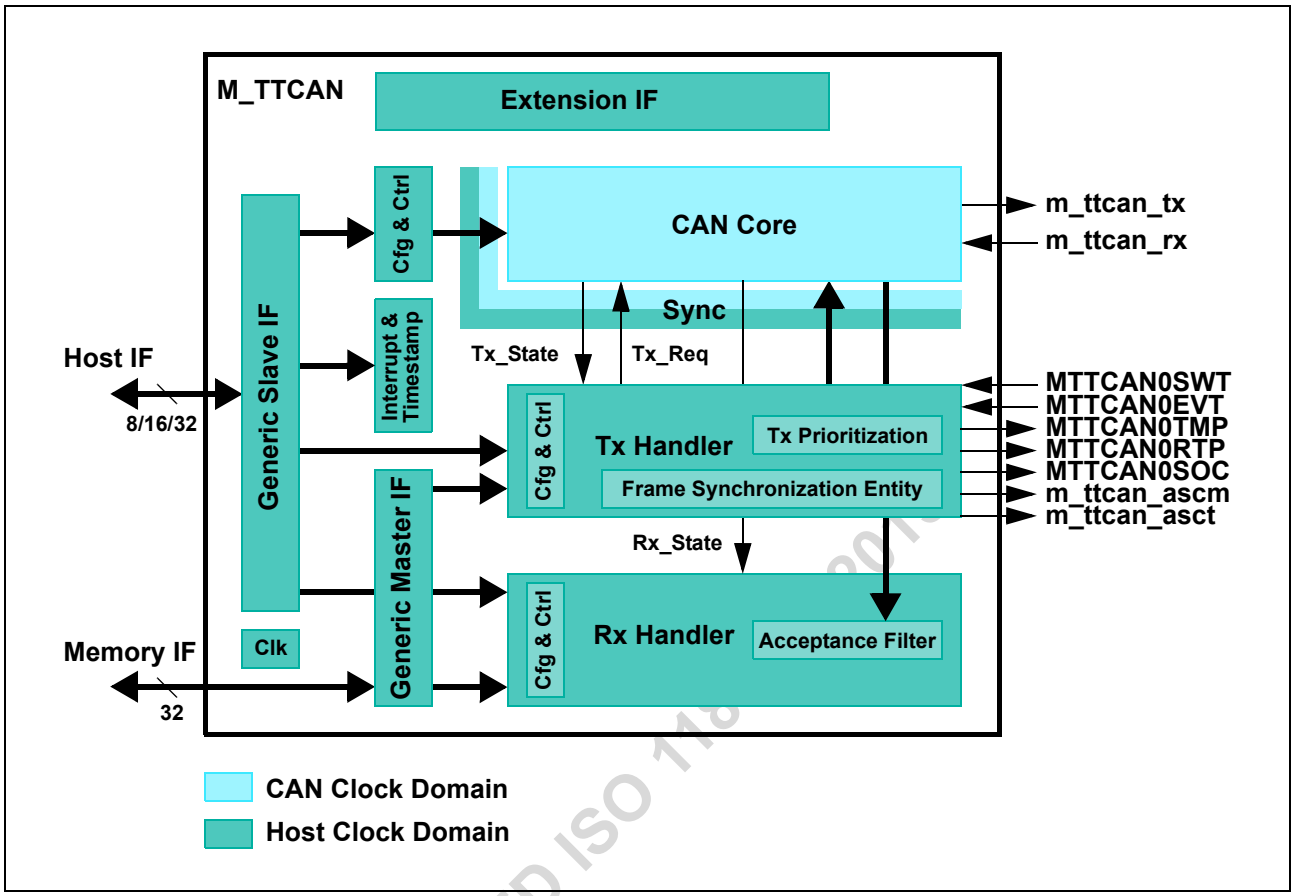



Figure 19.15 M_TTCAN Block Diagram

#### CAN Core

CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.

#### Sync

Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.

#### Clk

Synchronizes reset signal to the Host clock domain and to the CAN clock domain.

#### Cfg & Ctrl

CAN Core related configuration and control bits.

#### Interrupt & Timestamp

Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation. An externally generated 16-bit vector may substitute the integrated 16-bit CAN bit time counter for receive and transmit timestamp generation.

ISO CANFD ISO 11898-1:2015

### Tx Handler

Controls the message transfer from the external Message RAM to the CAN Core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.

The Tx Handler also implements the Frame Synchronization Entity FSE which controls time-triggered communication according to ISO11898-4. It synchronizes itself to the reference messages on the CAN bus, controls cycle time and global time, and handles transmissions according to the predefined message schedule, the system matrix. It also handles the time marks of the system matrix that are linked to the messages in the Message RAM. Stop Watch Trigger, Event Trigger, and Time Mark Interrupt are synchronization interfaces.

### Rx Handler

Controls the transfer of received messages from the CAN Core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

### Generic Slave Interface

Connects the M_TTCAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8/16/32-bit bus to support a wide range of interconnection structures.

### Generic Master Interface

Connects the M_TTCAN to a local 32-bit Message RAM. The implemented Message RAM size is 2K • 32 bit.

### Extension Interface

All flags from the Interrupt Register MTTCAN0IR and TT Interrupt Register MTTCAN0TTIR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M_TTCAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

ISO CANFD ISO 11898-1:2015

#### 19.6.1.3 Dual Clock Sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain m_ttcan_hclk (CLK_HSB). Due to the high precision clocking requirements of the CAN Core, a separate clock without any modulation has to be provided as m_ttcan_cclk (CLKP_H2).

Within the M_TTCAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

**NOTE**

In order to achieve a stable function of the M_TTCAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

#### 19.6.1.4 Dual Interrupt Lines

The module provides two interrupt lines. Interrupts can be routed either to m_ttcan_int0 (INTMTTCANI0) or to m_ttcan_int1(INTMTTCANI1). By default all interrupts are routed to interrupt line m_ttcan_int0 (INTMTTCANI0). By programming MTTCAN0ILE.EINT0 and MTTCAN0ILE.EINT1 the interrupt lines can be enabled or disabled separately.

ISO CANFD ISO 11898-1:2015

## 19.6.2 Programmer's Model

### 19.6.2.1 Hardware Reset Description

After hardware reset, the registers of the M_TTCAN hold the reset values listed in **Table 19.71**. Additionally the Bus_Off state is reset and the output m_ttcan_tx is set to *recessive* (HIGH). The value $0001_H$ (MTTCAN0CCCR.INIT = '1') in the CC Control Register enables software initialization. The M_TTCAN does not influence the CAN bus until the CPU resets MTTCAN0CCCR.INIT to '0'.

### 19.6.2.2 Register Map

The M_TTCAN module allocates an address space of 512 bytes. All registers are organized as 32-bit registers. The M_TTCAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8 bit (byte access), 16 bit (half-word access), or 32 bit (word access). Write access by the Host CPU to registers/bits marked with "P = Protected Write" is possible only with MTTCAN0CCCR.CCE = '1' AND MTTCAN0CCCR.INIT = '1'. There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

**Table 19.71 M_TTCAN Register Map (1/2)**

| ADDRESS | SYMBOL | NAME | PAGE | RESET | ACC |
|---|---|---|---|---|---|
| <MTTCAN0_base> + $000_H$ | MTTCAN0CREL | Core Release Register | 1097 | 3215 0323 | R |
| <MTTCAN0_base> + $004_H$ | MTTCAN0ENDN | Endian Register | 1098 | 8765 4321 | R |
| <MTTCAN0_base> + $00C_H$ | MTTCAN0DBTP | Data Bit Timing & Prescaler Register | 1099 | 0000 0A33 | RP |
| <MTTCAN0_base> + $010_H$ | MTTCAN0TEST | Test Register | 1101 | 0000 0000 | RP |
| <MTTCAN0_base> + $014_H$ | MTTCAN0RWD | RAM Watchdog | 1102 | 0000 0000 | RP |
| <MTTCAN0_base> + $018_H$ | MTTCAN0CCCR | CC Control Register | 1103 | 0000 0001 | RWPp |
| <MTTCAN0_base> + $01C_H$ | MTTCAN0NBTP | Nominal Bit Timing & Prescaler Register | 1105 | 0600 0A03 | RP |
| <MTTCAN0_base> + $020_H$ | MTTCAN0TSCC | Timestamp Counter Configuration | 1106 | 0000 0000 | RP |
| <MTTCAN0_base> + $024_H$ | MTTCAN0TSCV | Timestamp Counter Value | 1107 | 0000 0000 | RC |
| <MTTCAN0_base> + $028_H$ | MTTCAN0TOCC | Timeout Counter Configuration | 1108 | FFFF 0000 | RP |
| <MTTCAN0_base> + $02C_H$ | MTTCAN0TOCV | Timeout Counter Value | 1109 | 0000 FFFF | RC |
| <MTTCAN0_base> + $030_H$ to $03C_H$ | | reserved (4) | | 0000 0000 | R |
| <MTTCAN0_base> + $040_H$ | MTTCAN0ECR | Error Counter Register | 1110 | 0000 0000 | RX |
| <MTTCAN0_base> + $044_H$ | MTTCAN0PSR | Protocol Status Register | 1111 | 0000 0707 | RXS |
| <MTTCAN0_base> + $048_H$ | MTTCAN0TDCR | Transmitter Delay Compensation Register | 1111 | 0000 0000 | RP |
| <MTTCAN0_base> + $04C_H$ | | reserved (1) | | 0000 0000 | R |
| <MTTCAN0_base> + $050_H$ | MTTCAN0IR | Interrupt Register | 1114 | 0000 0000 | RW |
| <MTTCAN0_base> + $054_H$ | MTTCAN0IE | Interrupt Enable | 1117 | 0000 0000 | RW |
| <MTTCAN0_base> + $058_H$ | MTTCAN0ILS | Interrupt Line Select | 1119 | 0000 0000 | RW |
| <MTTCAN0_base> + $05C_H$ | MTTCAN0ILE | Interrupt Line Enable | 1121 | 0000 0000 | RW |
| <MTTCAN0_base> + $060_H$ to $07C_H$ | | reserved (8) | | 0000 0000 | R |
| <MTTCAN0_base> + $080_H$ | MTTCAN0GFC | Global Filter Configuration | 1122 | 0000 0000 | RP |
| <MTTCAN0_base> + $084_H$ | MTTCAN0SIDFC | Standard ID Filter Configuration | 1123 | 0000 0000 | RP |
| <MTTCAN0_base> + $088_H$ | MTTCAN0XIDFC | Extended ID Filter Configuration | 1124 | 0000 0000 | RP |
| <MTTCAN0_base> + $08C_H$ | | reserved (1) | | 0000 0000 | R |
| <MTTCAN0_base> + $090_H$ | MTTCAN0XIDAM | Extended ID AND Mask | 1125 | 1FFF FFFF | RP |
| <MTTCAN0_base> + $094_H$ | MTTCAN0HPMS | High Priority Message Status | 1126 | 0000 0000 | R |
| <MTTCAN0_base> + $098_H$ | MTTCAN0NDAT1 | New Data 1 | 1127 | 0000 0000 | RW |
| <MTTCAN0_base> + $09C_H$ | MTTCAN0NDAT2 | New Data 2 | 1128 | 0000 0000 | RW |
| <MTTCAN0_base> + $0A0_H$ | MTTCAN0RXF0C | Rx FIFO 0 Configuration | 1129 | 0000 0000 | RP |

ISO CAN FD ISO 11898-1:2015

Table 19.71 M_TTCAN Register Map (2/2)

| ADDRESS | SYMBOL | NAME | PAGE | RESET | ACC |
|---|---|---|---|---|---|
| <MTTCAN0_base> + $0A4_H$ | MTTCAN0RXF0S | Rx FIFO 0 Status | 1130 | 0000 0000 | R |
| <MTTCAN0_base> + $0A8_H$ | MTTCAN0RXF0A | Rx FIFO 0 Acknowledge | 1131 | 0000 0000 | RW |
| <MTTCAN0_base> + $0AC_H$ | MTTCAN0RXBC | Rx Buffer Configuration | 1132 | 0000 0000 | RP |
| <MTTCAN0_base> + $0B0_H$ | MTTCAN0RXF1C | Rx FIFO 1 Configuration | 1133 | 0000 0000 | RP |
| <MTTCAN0_base> + $0B4_H$ | MTTCAN0RXF1S | Rx FIFO 1 Status | 1134 | 0000 0000 | R |
| <MTTCAN0_base> + $0B8_H$ | MTTCAN0RXF1A | Rx FIFO 1 Acknowledge | 1135 | 0000 0000 | RW |
| <MTTCAN0_base> + $0BC_H$ | MTTCAN0RXESC | Rx Buffer / FIFO Element Size Configuration | 1136 | 0000 0000 | RP |
| <MTTCAN0_base> + $0C0_H$ | MTTCAN0TXBC | Tx Buffer Configuration | 1137 | 0000 0000 | RP |
| <MTTCAN0_base> + $0C4_H$ | MTTCAN0TXFQS | Tx FIFO/Queue Status | 1138 | 0000 0000 | R |
| <MTTCAN0_base> + $0C8_H$ | MTTCAN0TXESC | Tx Buffer Element Size Configuration | 1139 | 0000 0000 | RP |
| <MTTCAN0_base> + $0CC_H$ | MTTCAN0TXBRP | Tx Buffer Request Pending | 1140 | 0000 0000 | R |
| <MTTCAN0_base> + $0D0_H$ | MTTCAN0TXBAR | Tx Buffer Add Request | 1141 | 0000 0000 | RW |
| <MTTCAN0_base> + $0D4_H$ | MTTCAN0TXBCR | Tx Buffer Cancellation Request | 1142 | 0000 0000 | RW |
| <MTTCAN0_base> + $0D8_H$ | MTTCAN0TXBTO | Tx Buffer Transmission Occurred | 1143 | 0000 0000 | R |
| <MTTCAN0_base> + $0DC_H$ | MTTCAN0TXBCF | Tx Buffer Cancellation Finished | 1144 | 0000 0000 | R |
| <MTTCAN0_base> + $0E0_H$ | MTTCAN0TXBTIE | Tx Buffer Transmission Interrupt Enable | 1144 | 0000 0000 | RW |
| <MTTCAN0_base> + $0E4_H$ | MTTCAN0TXBCIE | Tx Buffer Cancellation Finished Interrupt Enable | 1145 | 0000 0000 | RW |
| <MTTCAN0_base> + $0E8_H$ to $0EC_H$ | | reserved (2) | | 0000 0000 | R |
| <MTTCAN0_base> + $0F0_H$ | MTTCAN0TXEFC | Tx Event FIFO Configuration | 1146 | 0000 0000 | RP |
| <MTTCAN0_base> + $0F4_H$ | MTTCAN0TXEFS | Tx Event FIFO Status | 1147 | 0000 0000 | R |
| <MTTCAN0_base> + $0F8_H$ | MTTCAN0TXEFA | Tx Event FIFO Acknowledge | 1148 | 0000 0000 | RW |
| <MTTCAN0_base> + $0FC_H$ | | reserved (1) | | 0000 0000 | R |
| <MTTCAN0_base> + $100_H$ | MTTCAN0TTTMC | TT Trigger Memory Configuration | 1149 | 0000 0000 | RP |
| <MTTCAN0_base> + $104_H$ | MTTCAN0TTRMC | TT Reference Message Configuration | 1150 | 0000 0000 | RP |
| <MTTCAN0_base> + $108_H$ | MTTCAN0TTOCF | TT Operation Configuration | 1151 | 0001 0000 | RP |
| <MTTCAN0_base> + $10C_H$ | MTTCAN0TTMLM | TT Matrix Limits | 1153 | 0000 0000 | RP |
| <MTTCAN0_base> + $110_H$ | MTTCAN0TURCF | TUR Configuration | 1154 | 1000 0000 | RP |
| <MTTCAN0_base> + $114_H$ | MTTCAN0TTOCN | TT Operation Control | 1156 | 0000 0000 | RW |
| <MTTCAN0_base> + $118_H$ | MTTCAN0TTGTP | TT Global Time Preset | 1158 | 0000 0000 | RW |
| <MTTCAN0_base> + $11C_H$ | MTTCAN0TTTMK | TT Time Mark | 1159 | 0000 0000 | RW |
| <MTTCAN0_base> + $120_H$ | MTTCAN0TTIR | TT Interrupt Register | 1160 | 0000 0000 | RW |
| <MTTCAN0_base> + $124_H$ | MTTCAN0TTIE | TT Interrupt Enable | 1162 | 0000 0000 | RW |
| <MTTCAN0_base> + $128_H$ | MTTCAN0TTILS | TT Interrupt Line Select | 1163 | 0000 0000 | RW |
| <MTTCAN0_base> + $12C_H$ | MTTCAN0TTOST | TT Operation Status | 1164 | 0000 0080 | R |
| <MTTCAN0_base> + $130_H$ | MTTCAN0TURNA | TUR Numerator Actual | 1166 | 0001 0000 | R |
| <MTTCAN0_base> + $134_H$ | MTTCAN0TTLGT | TT Local & Global Time | 1167 | 0000 0000 | R |
| <MTTCAN0_base> + $138_H$ | MTTCAN0TTCTC | TT Cycle Time & Count | 1168 | 003F 0000 | R |
| <MTTCAN0_base> + $13C_H$ | MTTCAN0TTCPT | TT Capture Time | 1169 | 0000 0000 | R |
| <MTTCAN0_base> + $140_H$ | MTTCAN0TTCSM | TT Cycle Sync Mark | 1170 | 0000 0000 | R |
| <MTTCAN0_base> + $144_H$ to $1FC_H$ | | reserved (47) | | 0000 0000 | R |

**Note:** R = Read, S = Set on read, X = Reset on read, W = Write, P = Protected write, p = Protected set, C = Clear/preset on write

ISO CANFD ISO 11898-1:2015

#### (1) Access to reserved Register Addresses

In case the application software wants to access one of the reserved addresses in the M_TTCAN register map (read or write access), interrupt flag MTTCAN0IR.ARA is set, and if enable the interrupt is signalled via the assigned interrupt line (m_ttcan_int0 or m_ttcan_int1).

### 19.6.2.3 Registers

#### (1) MTTCAN0CREL — Core Release Register

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $000_H$

**Value after reset:** $3215\ 0323_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | REL[3:0] | | | | STEP[3:0] | | | | SUBSTEP[3:0] | | | | YEAR[3:0] | | | |
| Value after reset | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MON[7:0] | | | | | | | | DAY[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

ISO CANFD ISO 11898-1:2015

**Table 19.72 MTTCAN0CREL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | REL[3:0] | Core Release<br>One digit, BCD-coded. |
| 27 to 24 | STEP[3:0] | Step of Core Release<br>One digit, BCD-coded. |
| 23 to 20 | SUBSTEP[3:0] | Sub-step of Core Release<br>One digit, BCD-coded. |
| 19 to 16 | YEAR[3:0] | Time Stamp Year<br>One digit, BCD-coded. This field is set by generic parameter on M_TTCAN synthesis. |
| 15 to 8 | MON[7:0] | Time Stamp Month<br>Two digits, BCD-coded. This field is set by generic parameter on M_TTCAN synthesis. |
| 7 to 0 | DAY[7:0] | Time Stamp Day<br>Two digits, BCD-coded. This field is set by generic parameter on M_TTCAN synthesis. |

**Table 19.73 Coding of Revisions**

| Release | Step | SubStep | Year | Month | Day | Name |
|---|---|---|---|---|---|---|
| 3 | 2 | 1 | 5 | 03 | 23 | Revision 3.2.1, Date 2015/03/23 |

### (2) MTTCAN0ENDN — Endian Register

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $004_H$

**Value after reset:** $8765\ 4321_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETV[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETV[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.74 MTTCAN0ENDN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ETV[31:0] | Endianness Test Value<br>The endianness test value is $8765\ 4321_H$. |

ISO CANFD ISO 11898-1:2015

### (3) MTTCAN0DBTP — Data Bit Timing & Prescaler Register

This register is only writable if bits MTTCAN0CCCR.CCE and MTTCAN0CCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 m_ttcan_cclk (CLKP_H2) periods. tq = (DBRP + 1) mtq.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [DTSEG1 + DTSEG2 + 3] tq
or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $00C_H$

**Value after reset:** $0000\ 0A33_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TDC | — | — | DBRP[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | RP | R | R | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DTSEG1[4:0] | | | | | DTSEG2[3:0] | | | | DSJW[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

ISO CAN FD ISO 11898-1:2015

**Table 19.75 MTTCAN0DBTP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 | TDC | Transmitter Delay Compensation<br>0: Transmitter Delay Compensation disabled<br>1: Transmitter Delay Compensation enabled |
| 22, 21 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 20 to 16 | DBRP[4:0] | Data Bit Rate Prescaler<br>$00_H$ to $1F_H$<br>The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Bit Rate Prescaler are 0 to 31. When TDC = 1, the range is limited to 0, 1. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 15 to 13 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 12 to 8 | DTSEG1[4:0] | Data time segment before sample point<br>$00_H$ to $1F_H$<br>Valid values are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 7 to 4 | DTSEG2[3:0] | Data time segment after sample point<br>$0_H$ to $F_H$<br>Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 3 to 0 | DSJW[3:0] | Data (Re) Synchronization Jump Width<br>$0_H$ to $F_H$<br>Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |

**NOTES**

1. With a CAN clock (m_ttcan_cclk: CLKP_H2) of 8 MHz, the reset value of 0000 0A33H configures the M_TTCAN for a data phase bit rate of 500 kBit/s.
2. The bit rate configured for the CAN FD data phase via MTTCAN0DBTP must be higher or equal to the bit rate configured for the arbitration phase via MTTCAN0NBTP.

ISO CANFD ISO 11898-1:2015

### (4) MTTCAN0TEST — Test Register

Write access to the Test Register has to be enabled by setting bit MTTCAN0CCCR.TEST to '1'. All Test Register functions are set to their reset values when bit MTTCAN0CCCR.TEST is reset.

Loop Back Mode and software control of pin m_ttcan_tx are hardware test modes. Programming of TX ≠ "00" may disturb the message transfer on the CAN bus.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $010_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RX | TX[1:0] | | LBCK | CAT | CAM | TAT | TAM |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | RP | RP | RP | R | R | RP | RP |

**Table 19.76 MTTCAN0TEST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 7 | RX | Receive Pin<br>Monitors the actual value of pin m_ttcan_rx<br>0: The CAN bus is dominant (m_ttcan_rx = '0').<br>1: The CAN bus is recessive (m_ttcan_rx = '1'). |
| 6, 5 | TX[1:0] | Control of Transmit Pin<br>00: Reset value, m_ttcan_tx controlled by the CAN Core, updated at the end of the CAN bit time.<br>01: Sample Point can be monitored at pin m_ttcan_tx.<br>10: Dominant ('0') level at pin m_ttcan_tx<br>11: Recessive ('1') at pin m_ttcan_tx |
| 4 | LBCK | Loop Back Mode<br>0: Reset value, Loop Back Mode is disabled.<br>1: Loop Back Mode is enabled (see **(9) Test Modes**). |
| 3 | CAT | Check ASC Transmit Control<br>Monitors level at output pin m_ttcan_asct.<br>0: Output pin m_ttcan_asct = '0'<br>1: Output pin m_ttcan_asct = '1' |
| 2 | CAM | Check ASC Multiplexer Control<br>Monitors level at output pin m_ttcan_ascm.<br>0: Output pin m_ttcan_ascm = '0'<br>1: Output pin m_ttcan_ascm = '1' |
| 1 | TAT | Test ASC Transmit Control<br>Controls output pin m_ttcan_asct in test mode, ORed with the signal from the FSE<br>0: Level at pin m_ttcan_asct controlled by FSE<br>1: Level at pin m_ttcan_asct = '1' |
| 0 | TAM | Test ASC Multiplexer Control<br>Controls output pin m_ttcan_ascm in test mode, ORed with the signal from the FSE<br>0: Level at pin m_ttcan_ascm controlled by FSE<br>1: Level at pin m_ttcan_ascm = '1' |

ISO CANFD 11898-1:2015

### (5) MTTCAN0RWD — RAM Watchdog

The RAM Watchdog monitors the READY output of the Message RAM (m_ttcan_aeim_ready). A Message RAM access via the M_TTCAN's Generic Master Interface (m_ttcan_aeim_sel active) starts the Message RAM Watchdog Counter with the value configured by MTTCAN0RWD.WDC. The counter is reloaded with MTTCAN0RWD.WDC when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag MTTCAN0IR.WDI is set. The RAM Watchdog Counter is clocked by the Host clock (m_ttcan_hclk: CLK_HSB).

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $014_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WDV[7:0] | | | | | | | | WDC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP |

**Table 19.77 MTTCAN0RWD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 8 | WDV[7:0] | Watchdog Value<br>Actual Message RAM Watchdog Counter Value. |
| 7 to 0 | WDC[7:0] | Watchdog Configuration<br>Start value of the Message RAM Watchdog Counter. With the reset value of "00" the counter is disabled. |

ISO CANFD CO 11898-1:2015

### (6) MTTCAN0CCCR — CC Control Register

For details about setting and resetting of single bits see **(1) Software Initialization**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $018_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NISO | TXP | EFBI | PXHD | — | — | BRSE | FDOE | TEST | DAR | MON | CSR | CSA | ASM | CCE | INIT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | RP | RP | RP | RP | R | R | RP | RP | Rp | RP | Rp | R/W | R | Rp | RP | R/W |

**Table 19.78 MTTCAN0CCCR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 | NISO | Non ISO Operation<br>If this bit is set, the M_TTCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.<br>0: CAN FD frame format according to ISO11898-1:2015<br>1: CAN FD frame format according to Bosch CAN FD Specification V1.0 |
| 14 | TXP | Transmit Pause<br>If this bit is set, the M_TTCAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame (see **(1) Transmit Pause**).<br>0: Transmit pause disabled<br>1: Transmit pause enabled |
| 13 | EFBI | Edge Filtering during Bus Integration<br>0: Edge filtering disabled.<br>1: Two consecutive dominant tq required to detect an edge for hard synchronization. |
| 12 | PXHD | Protocol Exception Handling Disable<br>0: Protocol exception handling enabled.<br>1: Protocol exception handling disabled.<br>**NOTE**<br>When protocol exception handling is disabled, the M_TTCAN will transmit an error frame when it detects a protocol exception condition. |
| 11, 10 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 9 | BRSE | Bit Rate Switch Enable<br>0: Bit rate switching for transmissions disabled.<br>1: Bit rate switching for transmissions enabled.<br>**NOTE**<br>When CAN FD operation is disabled FDOE = '0', BRSE is not evaluated. |
| 8 | FDOE | FD Operation Enable<br>0: FD operation disabled.<br>1: FD operation enabled. |

ISO CANFD IP Conformance 2015

**Table 19.78 MTTCAN0CCCR Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | TEST | Test Mode Enable<br>0: Normal operation, register MTTCAN0TEST holds reset values.<br>1: Test Mode, write access to register MTTCAN0TEST enabled. |
| 6 | DAR | Disable Automatic Retransmission<br>0: Automatic retransmission of messages not transmitted successfully enabled.<br>1: Automatic retransmission disabled |
| 5 | MON | Bus Monitoring Mode<br>Bit MON can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time.<br>0: Bus Monitoring Mode is disabled<br>1: Bus Monitoring Mode is enabled |
| 4 | CSR | Clock Stop Request<br>0: No clock stop is requested.<br>1: Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle. |
| 3 | CSA | Clock Stop Acknowledge<br>0: No clock stop acknowledged.<br>1: M_TTCAN may be set in power down by stopping m_ttcan_hclk (CLK_HSB) and m_ttcan_cclk (CLKP_H2). |
| 2 | ASM | Restricted Operation Mode<br>Bit ASM can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. For a description of the Restricted Operation Mode see **(5) Restricted Operation Mode**.<br>0: Normal CAN operation<br>1: Restricted Operation Mode active |
| 1 | CCE | Configuration Change Enable<br>0: The CPU has no write access to the protected configuration registers.<br>1: The CPU has write access to the protected configuration registers (while MTTCAN0CCCR.INIT = '1'). |
| 0 | INIT | Initialization<br>0: Normal Operation<br>1: Initialization is started. |

**NOTE**

Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

NEO CANFD ISO 11898-1:2015

### (7) MTTCAN0NBTP — Nominal Bit Timing & Prescaler Register

This register is only writable if bits MTTCAN0CCCR.CCE and MTTCAN0CCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 385 time quanta. The CAN time quantum may be programmed in the range of 1 to 512 m_ttcan_cclk (CLKP_H2) periods. tq = (NBRP + 1) mtq.

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [NTSEG1 + NTSEG2 + 3] tq
or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $01C_H$

**Value after reset:** $0600\ 0A03_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NSJW[6:0] | | | | | | | NBRP[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NTSEG1[7:0] | | | | | | | | — | NTSEG2[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

ISO CAN FD ISO 11898-1:2015

**Table 19.79 MTTCAN0NBTP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 25 | NSJW[6:0] | Nominal (Re)Synchronization Jump Width.<br>$00_H$ to $7F_H$<br>Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 24 to 16 | NBRP[8:0] | Nominal Bit Rate Prescaler<br>$000_H$ to $1FF_H$<br>The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Bit Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 15 to 8 | NTSEG1[7:0] | Nominal Time segment before sample point<br>$01_H$ to $FF_H$<br>Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 7 | Reserved | These bits are always read as 0. |
| 6 to 0 | NTSEG2[6:0] | Nominal Time segment after sample point<br>$00_H$ to $7F_H$<br>Valid values are 1 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |

**NOTE**

With a CAN clock (m_ttcan_cclk: CLKP_H2) of 8 MHz, the reset value of $0600\ 0A03_H$ configures the M_TTCAN for a bit rate of 500 kBit/s.

### (8) MTTCAN0TSCC — Timestamp Counter Configuration

For a description of the Timestamp Counter see **Section 19.6.3.2, Timestamp Generation**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $020_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TCP[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TSS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP |

**Table 19.80 MTTCAN0TSCC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 19 to 16 | TCP[3:0] | Timestamp Counter Prescaler<br>$0_H$ to $F_H$<br>Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 to 16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.<br>**NOTE**<br>With CAN FD an external counter is required for timestamp generation (TSS = “10”) |
| 15 to 2 | Reserved | These bits are always read as 0. |
| 1, 0 | TSS[1:0] | Timestamp Select<br>00: Timestamp counter value always $0000_H$<br>01: Timestamp counter value incremented according to TCP<br>10: External timestamp counter value used<br>11: Same as “00” |

ISO CANFD IP, 1.0 - 1-2015

### (9) MTTCAN0TSCV — Timestamp Counter Value

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $024_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSC[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

**Table 19.81 MTTCAN0TSCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 0 | TSC[15:0] | Timestamp Counter<br>The internal/external Timestamp Counter value is captured on start of frame (both Rx and Tx). When MTTCAN0TSCC.TSS = “01”, the Timestamp Counter is incremented in multiples of CAN bit times [1 to 16] depending on the configuration of MTTCAN0TSCC.TCP. A wrap around sets interrupt flag MTTCAN0IR.TSW. Write access resets the counter to zero. When MTTCAN0TSCC.TSS = “10”, TSC reflects the external Timestamp Counter value. A write access has no impact. |

**NOTE**

A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by write access to MTTCAN0TSCV.

ISO CANFD ISO 11898-1:2015

### (10) MTTCAN0TOCC — Timeout Counter Configuration

For a description of the Timeout Counter see **Section 19.6.3.3, Timeout Counter**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $028_H$

**Value after reset:** FFFF $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TOP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TOS[1:0] | | ETOC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP |

**Table 19.82 MTTCAN0TOCC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TOP[15:0] | Timeout Period<br>Start value of the Timeout Counter (down-counter). Configures the Timeout Period. |
| 15 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2, 1 | TOS[1:0] | Timeout Select<br>When operating in Continuous mode, a write to MTTCAN0TOCV presets the counter to the value configured by MTTCAN0TOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by MTTCAN0TOCC.TOP. Down-counting is started when the first FIFO element is stored.<br>00: Continuous operation<br>01: Timeout controlled by Tx Event FIFO<br>10: Timeout controlled by Rx FIFO 0<br>11: Timeout controlled by Rx FIFO 1 |
| 0 | ETOC | Enable Timeout Counter<br>0: Timeout Counter disabled<br>1: Timeout Counter enabled |

**NOTE**

For use of timeout function with CAN FD see **Section 19.5.3.3, Timeout Counter**.

ISO CANFD ISO 11898-1:2015

### (11) MTTCAN0TOCV — Timeout Counter Value

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $02C_H$

**Value after reset:** $0000\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TOC[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

**Table 19.83 MTTCAN0TOCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 0 | TOC[15:0] | Timeout Counter<br>The Timeout Counter is decremented in multiples of CAN bit times [1 to 16] depending on the configuration of MTTCAN0TSCC.TCP. When decremented to zero, interrupt flag MTTCAN0IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via MTTCAN0TOCC.TOS. |

ISO CANFD ISO 11898-1:2015

### (12) MTTCAN0ECR — Error Counter Register

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $040_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CEL[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | X | X | X | X | X | X | X | X |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RP | REC[6:0] | | | | | | | TEC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.84 MTTCAN0ECR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. |
| 23 to 16 | CEL[7:0] | CAN Error Logging<br>The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at $FF_H$; the next increment of TEC or REC sets interrupt flag MTTCAN0IR.ELO. |
| 15 | RP | Receive Error Passive<br>0: The Receive Error Counter is below the error passive level of 128<br>1: The Receive Error Counter has reached the error passive level of 128 |
| 14 to 8 | REC[6:0] | Receive Error Counter<br>Actual state of the Receive Error Counter, values between 0 and 127 |
| 7 to 0 | TEC[7:0] | Transmit Error Counter<br>Actual state of the Transmit Error Counter, values between 0 and 255 |

**NOTE**

When MTTCAN0CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented. This enables monitoring of collisions between CAN frames and ASC frames.

ISO CANFD ISO 11898-1:2015

### (13) MTTCAN0PSR — Protocol Status Register

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $044_H$

**Value after reset:** 0000 $0707_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | TDCV[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | PXE | RFDF | RBRS | RESI | DLEC[2:0] | | | BO | EW | EP | ACT[1:0] | | LEC[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| R/W | R | X | X | X | X | S | S | S | R | R | R | R | R | S | S | S |

**Table 19.85 MTTCAN0PSR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | These bits are always read as 0. |
| 22 to 16 | TDCV[6:0] | Transmitter Delay Compensation Value<br>$00_H$ to $7F_H$<br>Position of the secondary sample point, defined by the sum of the measured delay from m_ttcan_tx to m_ttcan_rx and MTTCAN0TDCR.TDCO. The SSP position is, in the data phase, the number of mtq between the start of the transmitted bit and the secondary sample point Valid values are 0 to 127 mtq. |
| 14 | PXE | Protocol Exception Event<br>0: No protocol exception event occurred since last read access.<br>1: Protocol exception event occurred. |
| 13 | RFDF | Received a CAN FD Message<br>This bit is set independent of acceptance filtering.<br>0: Since this bit was reset by the CPU, no CAN FD message has been received<br>1: Message in CAN FD format with FDF flag set has been received |
| 12 | RBRS | BRS flag of last received CAN FD Message<br>This bit is set together with RFDF, independent of acceptance filtering.<br>0: Last received CAN FD message did not have its BRS flag set<br>1: Last received CAN FD message had its BRS flag set |
| 11 | RESI | ESI flag of last received CAN FD Message<br>This bit is set together with RFDF, independent of acceptance filtering.<br>0: Last received CAN FD message did not have its ESI flag set<br>1: Last received CAN FD message had its ESI flag set |
| 10 to 8 | DLEC[2:0] | Data Phase Last Error Code<br>Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. |
| 7 | BO | Bus_Off Status<br>0: The M_TTCAN is not Bus_Off<br>1: The M_TTCAN is in Bus_Off state |
| 6 | EW | Warning Status<br>0: Both error counters are below the Error_Warning limit of 96<br>1: At least one of error counter has reached the Error_Warning limit of 96 |

ISO CANFD ISO 11898-1:2015

**Table 19.85 MTTCAN0PSR Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 5 | EP | Error Passive<br>0: The M_TTCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected<br>1: The M_TTCAN is in the Error_Passive state |
| 4, 3 | ACT[1:0] | Activity<br>Monitors the module's CAN communication state.<br>00: Synchronizing - node is synchronizing on CAN communication<br>01: Idle - node is neither receiver nor transmitter<br>10: Receiver - node is operating as receiver<br>11: Transmitter - node is operating as transmitter<br>**NOTE**<br>ACT is set to "00" by a Protocol Exception Event. |
| 2 to 0 | LEC[2:0] | Last Error Code<br>The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error.<br>0: No Error: No error occurred since LEC has been reset by successful reception or transmission.<br>1: Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.<br>2: Form Error: A fixed format part of a received frame has the wrong format.<br>3: AckError: The message transmitted by the M_TTCAN was not acknowledged by another node.<br>4: Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.<br>5: Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).<br>6: CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.<br>7: NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register. |

ISO CANFD 16.08.2015

**NOTES**

1. When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in DLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.
2. The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1:2015) cannot be shortened by setting or resetting MTTCAN0CCCR.INIT. If the device goes Bus_Off, it will set MTTCAN0CCCR.INIT of its own accord, stopping all bus activities. Once MTTCAN0CCCR.INIT has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of MTTCAN0CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to MTTCAN0PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. MTTCAN0ECR.REC is used to count these sequences.

### (14) MTTCAN0TDCR — Transmitter Delay Compensation Register

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $048_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TDCO[6:0] | | | | | | | — | TDCF[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

**Table 19.86 MTTCAN0TDCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 15 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 14 to 8 | TDCO[6:0] | Transmitter Delay Compensation Offset<br>$00_H$ to $7F_H$<br>Offset value defining the distance between the measured delay from m_ttcan_tx to m_ttcan_rx and the secondary sample point. Valid values are 0 to 127 mtq. |
| 7 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 6 to 0 | TDCF[6:0] | Transmitter Delay Compensation Filter Window Length<br>$00_H$ to $7F_H$<br>Defines the minimum value for the SSP position, dominant edges on m_ttcan_rx that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCO. Valid values are 0 to 127 mtq. |

ISO CANFD ISO 11898-1:2015

### (15) MTTCAN0IR — Interrupt Register

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. The configuration of MTTCAN0IE controls whether an interrupt is generated. The configuration of MTTCAN0ILS controls on which interrupt line an interrupt is signalled.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $050_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | ARA | PED | PEA | WDI | BO | EW | EP | ELO | BEU | BEC | DRX | TOO | MRAF | TSW |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFL | TEFF | TEFW | TEFN | TFE | TCF | TC | HPM | RF1L | RF1F | RF1W | RF1N | RF0L | RF0F | RF0W | RF0N |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

ISO CANFD IP 11898-1:2015

**Table 19.87 MTTCAN0IR Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 29 | ARA | Access to Reserved Address<br>0: No access to reserved address occurred<br>1: Access to reserved address occurred |
| 28 | PED | Protocol Error in Data Phase (Data Bit Time is used)<br>0: No protocol error in data phase<br>1: Protocol error in data phase detected (MTTCAN0PSR.DLEC ≠ 0,7) |
| 27 | PEA | Protocol Error in Arbitration Phase (Nominal Bit Time is used)<br>0: No protocol error in arbitration phase<br>1: Protocol error in arbitration phase detected(MTTCAN0PSR.LEC ≠ 0,7) |
| 26 | WDI | Watchdog Interrupt<br>0: No Message RAM Watchdog event occurred<br>1: Message RAM Watchdog event due to missing READY |
| 25 | BO | Bus_Off Status<br>0: Bus_Off status unchanged<br>1: Bus_Off status changed |
| 24 | EW | Warning Status<br>0: Error_Warning status unchanged<br>1: Error_Warning status changed |
| 23 | EP | Error Passive<br>0: Error_Passive status unchanged<br>1: Error_Passive status changed |
| 22 | ELO | Error Logging Overflow<br>0: CAN Error Logging Counter did not overflow<br>1: Overflow of CAN Error Logging Counter occurred |

Table 19.87 MTTCAN0IR Register Contents (2/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 21 | BEU | Bit Error Uncorrected<br>Message RAM bit error detected, uncorrected. Controlled by input signal m_ttcan_aeim_berr[1] generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets MTTCAN0CCCR.INIT to '1'. This is done to avoid transmission of corrupted data.<br>0: No bit error detected when reading from Message RAM<br>1: Bit error detected, uncorrected (e.g. parity logic) |
| 20 | BEC | Bit Error Corrected<br>Message RAM bit error detected and corrected. Controlled by input signal m_ttcan_aeim_berr[0] generated by an optional external parity / ECC logic attached to the Message RAM.<br>0: No bit error detected when reading from Message RAM<br>1: Bit error detected and corrected (e.g. ECC) |
| 19 | DRX | Message stored to Dedicated Rx Buffer<br>The flag is set whenever a received message has been stored into a dedicated Rx Buffer.<br>0: No Rx Buffer updated<br>1: At least one received message stored into an Rx Buffer |
| 18 | TOO | Timeout Occurred<br>0: No timeout<br>1: Timeout reached |
| 17 | MRAF | Message RAM Access Failure<br>The flag is set, when the Rx Handler<br>• has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message.<br>• was not able to write a message to the Message RAM. In this case message storage is aborted.<br>In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location.<br>The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the M_TTCAN is switched into Restricted Operation Mode (see **(5) Restricted Operation Mode**). To leave Restricted Operation Mode, the Host CPU has to reset MTTCAN0CCCR.ASM.<br>0: No Message RAM access failure occurred<br>1: Message RAM access failure occurred |
| 16 | TSW | Timestamp Wraparound<br>0: No timestamp counter wrap-around<br>1: Timestamp counter wrapped around |
| 15 | TEFL | Tx Event FIFO Element Lost<br>0: No Tx Event FIFO element lost<br>1: Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero |
| 14 | TEFF | Tx Event FIFO Full<br>0: Tx Event FIFO not full<br>1: Tx Event FIFO full |
| 13 | TEFW | Tx Event FIFO Watermark Reached<br>0: Tx Event FIFO fill level below watermark<br>1: Tx Event FIFO fill level reached watermark |
| 12 | TEFN | Tx Event FIFO New Entry<br>0: Tx Event FIFO unchanged<br>1: Tx Handler wrote Tx Event FIFO element |
| 11 | TFE | Tx FIFO Empty<br>0: Tx FIFO non-empty<br>1: Tx FIFO empty |

ISO CANFD ISO 11898-1:2015

Table 19.87 MTTCAN0IR Register Contents (3/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 10 | TCF | Transmission Cancellation Finished<br>0: No transmission cancellation finished<br>1: Transmission cancellation finished |
| 9 | TC | Transmission Completed<br>0: No transmission completed<br>1: Transmission completed |
| 8 | HPM | High Priority Message<br>0: No high priority message received<br>1: High priority message received |
| 7 | RF1L | Rx FIFO 1 Message Lost<br>0: No Rx FIFO 1 message lost<br>1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero |
| 6 | RF1F | Rx FIFO 1 Full<br>0: Rx FIFO 1 not full<br>1: Rx FIFO 1 full |
| 5 | RF1W | Rx FIFO 1 Watermark Reached<br>0: Rx FIFO 1 fill level below watermark<br>1: Rx FIFO 1 fill level reached watermark |
| 4 | RF1N | Rx FIFO 1 New Message<br>0: No new message written to Rx FIFO 1<br>1: New message written to Rx FIFO 1 |
| 3 | RF0L | Rx FIFO 0 Message Lost<br>0: No Rx FIFO 0 message lost<br>1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero |
| 2 | RF0F | Rx FIFO 0 Full<br>0: Rx FIFO 0 not full<br>1: Rx FIFO 0 full |
| 1 | RF0W | Rx FIFO 0 Watermark Reached<br>0: Rx FIFO 0 fill level below watermark<br>1: Rx FIFO 0 fill level reached watermark |
| 0 | RF0N | Rx FIFO 0 New Message<br>0: No new message written to Rx FIFO 0<br>1: New message written to Rx FIFO 0 |

ISO CANFD CC (1889-10-2015

### (16) MTTCAN0IE — Interrupt Enable

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signalled on an interrupt line.

0: Interrupt disabled

1: Interrupt enabled

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $054_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | ARAE | PEDE | PEAE | WDIE | BOE | EWE | EPE | ELOE | BEUE | BECE | DRXE | TOOE | MRAFE | TSWE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFLE | TEFFE | TEFWE | TEFNE | TFEE | TCFE | TCE | HPME | RF1LE | RF1FE | RF1WE | RF1NE | RF0LE | RF0FE | RF0WE | RF0NE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

ISO CANFD (ISO 11898-1:2015)

**Table 19.88 MTTCAN0IE Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 29 | ARAE | Access to Reserved Address Enable |
| 28 | PEDE | Protocol Error in Data Phase Enable |
| 27 | PEAE | Protocol Error in Arbitration Phase Enable |
| 26 | WDIE | Watchdog Interrupt Enable |
| 25 | BOE | Bus_Off Status Interrupt Enable |
| 24 | EWE | Warning Status Interrupt Enable |
| 23 | EPE | Error Passive Interrupt Enable |
| 22 | ELOE | Error Logging Overflow Interrupt Enable |
| 21 | BEUE | Bit Error Uncorrected Interrupt Enable |
| 20 | BECE | Bit Error Corrected Interrupt Enable |
| 19 | DRXE | Message stored to Dedicated Rx Buffer Interrupt Enable |
| 18 | TOOE | Timeout Occurred Interrupt Enable |
| 17 | MRAFE | Message RAM Access Failure Interrupt Enable |
| 16 | TSWE | Timestamp Wraparound Interrupt Enable |
| 15 | TEFLE | Tx Event FIFO Event Lost Interrupt Enable |
| 14 | TEFFE | Tx Event FIFO Full Interrupt Enable |
| 13 | TEFWE | Tx Event FIFO Watermark Reached Interrupt Enable |
| 12 | TEFNE | Tx Event FIFO New Entry Interrupt Enable |
| 11 | TFEE | Tx FIFO Empty Interrupt Enable |
| 10 | TCFE | Transmission Cancellation Finished Interrupt Enable |
| 9 | TCE | Transmission Completed Interrupt Enable |
| 8 | HPME | High Priority Message Interrupt Enable |
| 7 | RF1LE | Rx FIFO 1 Message Lost Interrupt Enable |

Table 19.88 MTTCAN0IE Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 | RF1FE | Rx FIFO 1 Full Interrupt Enable |
| 5 | RF1WE | Rx FIFO 1 Watermark Reached Interrupt Enable |
| 4 | RF1NE | Rx FIFO 1 New Message Interrupt Enable |
| 3 | RF0LE | Rx FIFO 0 Message Lost Interrupt Enable |
| 2 | RF0FE | Rx FIFO 0 Full Interrupt Enable |
| 1 | RF0WE | Rx FIFO 0 Watermark Reached Interrupt Enable |
| 0 | RF0NE | Rx FIFO 0 New Message Interrupt Enable |

ISO CANFD ISO 11898-1:2015

### (17) MTTCAN0ILS — Interrupt Line Select

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via MTTCAN0ILE.EINT0 and MTTCAN0ILE.EINT1.

0: Interrupt assigned to interrupt line m_ttcan_int0 (INTMTTCANI0)

1: Interrupt assigned to interrupt line m_ttcan_int1 (INTMTTCANI1)

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $058_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | ARAL | PEDL | PEAL | WDIL | BOL | EWL | EPL | ELOL | BEUL | BECL | DRXL | TOOL | MRAFL | TSWL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFLL | TEFFL | TEFWL | TEFNL | TFEL | TCFL | TCL | HPML | RF1LL | RF1FL | RF1WL | RF1NL | RF0LL | RF0FL | RF0WL | RF0NL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

ISO CANFD FCO 11898-1:2015

**Table 19.89 MTTCAN0ILS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 29 | ARAL | Access to Reserved Address Line |
| 28 | PEDL | Protocol Error in Data Phase Line |
| 27 | PEAL | Protocol Error in Arbitration Phase Line |
| 26 | WDIL | Watchdog Interrupt Line |
| 25 | BOL | Bus_Off Status Interrupt Line |
| 24 | EWL | Warning Status Interrupt Line |
| 23 | EPL | Error Passive Interrupt Line |
| 22 | ELOL | Error Logging Overflow Interrupt Line |
| 21 | BEUL | Bit Error Uncorrected Interrupt Line |
| 20 | BECL | Bit Error Corrected Interrupt Line |
| 19 | DRXL | Message stored to Dedicated Rx Buffer Interrupt Line |
| 18 | TOOL | Timeout Occurred Interrupt Line |
| 17 | MRAFL | Message RAM Access Failure Interrupt Line |
| 16 | TSWL | Timestamp Wraparound Interrupt Line |
| 15 | TEFLL | Tx Event FIFO Event Lost Interrupt Line |
| 14 | TEFFL | Tx Event FIFO Full Interrupt Line |
| 13 | TEFWL | Tx Event FIFO Watermark Reached Interrupt Line |
| 12 | TEFNL | Tx Event FIFO New Entry Interrupt Line |
| 11 | TFEL | Tx FIFO Empty Interrupt Line |
| 10 | TCFL | Transmission Cancellation Finished Interrupt Line |
| 9 | TCL | Transmission Completed Interrupt Line |
| 8 | HPML | High Priority Message Interrupt Line |

Table 19.89 MTTCAN0ILS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RF1LL | Rx FIFO 1 Message Lost Interrupt Line |
| 6 | RF1FL | Rx FIFO 1 Full Interrupt Line |
| 5 | RF1WL | Rx FIFO 1 Watermark Reached Interrupt Line |
| 4 | RF1NL | Rx FIFO 1 New Message Interrupt Line |
| 3 | RF0LL | Rx FIFO 0 Message Lost Interrupt Line |
| 2 | RF0FL | Rx FIFO 0 Full Interrupt Line |
| 1 | RF0WL | Rx FIFO 0 Watermark Reached Interrupt Line |
| 0 | RF0NL | Rx FIFO 0 New Message Interrupt Line |

ISO CANFD ISO 11898-1:2015

### (18) MTTCAN0ILE — Interrupt Line Enable

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $05C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EINT1 | EINT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 19.90 MTTCAN0ILE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1 | EINT1 | Enable Interrupt Line 1<br>0: Interrupt line m_ttcan_int1 (INTMTTCANI1) disabled<br>1: Interrupt line m_ttcan_int1 (INTMTTCANI1) enabled |
| 0 | EINT0 | Enable Interrupt Line 0<br>0: Interrupt line m_ttcan_int0 (INTMTTCANI0) disabled<br>1: Interrupt line m_ttcan_int0 (INTMTTCANI0) enabled |

ISO CANFD (ISO 11898-1:2015

### (19) MTTCAN0GFC — Global Filter Configuration

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in **Figure 19.20, Standard Message ID Filter Path** and **Figure 19.21, Extended Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $080_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ANFS[1:0] | | ANFE[1:0] | | RRFS | RRFE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP |

**Table 19.91 MTTCAN0GFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5, 4 | ANFS[1:0] | Accept Non-matching Frames Standard<br>Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.<br>00: Accept in Rx FIFO 0<br>01: Accept in Rx FIFO 1<br>10: Reject<br>11: Reject |
| 3, 2 | ANFE[1:0] | Accept Non-matching Frames Extended<br>Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.<br>00: Accept in Rx FIFO 0<br>01: Accept in Rx FIFO 1<br>10: Reject<br>11: Reject |
| 1 | RRFS | Reject Remote Frames Standard<br>0: Filter remote frames with 11-bit standard IDs<br>1: Reject all remote frames with 11-bit standard IDs |
| 0 | RRFE | Reject Remote Frames Extended<br>0: Filter remote frames with 29-bit extended IDs<br>1: Reject all remote frames with 29-bit extended IDs |

ISO CANFD ISO 11898-1:2015

### (20) MTTCAN0SIDFC — Standard ID Filter Configuration

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in **Figure 19.20, Standard Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $084_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | LSS[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLSSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.92 MTTCAN0SIDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 16 | LSS[7:0] | List Size Standard<br>0: No standard Message ID filter<br>1 to 128: Number of standard Message ID filter elements<br>>128: Values greater than 128 are interpreted as 128 |
| 15 to 2 | FLSSA[15:2] | Filter List Standard Start Address<br>Start address of standard Message ID filter list (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

ISO CANFD ICD Ver.08-1:2015

### (21) MTTCAN0XIDFC — Extended ID Filter Configuration

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in **Figure 19.21, Extended Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $088_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | LSE[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLESA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.93 MTTCAN0XIDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 22 to 16 | LSE[6:0] | List Size Extended<br>0: No extended Message ID filter<br>1 to 64: Number of extended Message ID filter elements<br>>64: Values greater than 64 are interpreted as 64 |
| 15 to 2 | FLESA[15:2] | Filter List Extended Start Address<br>Start address of extended Message ID filter list (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

ISO CANFD FDIS 11898-1:2015

### (22) MTTCAN0XIDAM — Extended ID AND Mask

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $090_H$

**Value after reset:** $1FFF\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | EIDM[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EIDM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

**Table 19.94 MTTCAN0XIDAM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 28 to 0 | EIDM[28:0] | Extended ID Mask<br>For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active. |

ISO CANFD ISO 11898-1:2015

### (23) MTTCAN0HPMS — High Priority Message Status

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $094_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLST | FIDX[6:0] | | | | | | | MSI[1:0] | | BIDX[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.95 MTTCAN0HPMS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. |
| 15 | FLST | Filter List<br>Indicates the filter list of the matching filter element.<br>0: Standard Filter List<br>1: Extended Filter List |
| 14 to 8 | FIDX[6:0] | Filter Index<br>Index of matching filter element. Range is 0 to MTTCAN0SIDFC.LSS – 1 resp. MTTCAN0XIDFC.LSE – 1. |
| 7, 6 | MSI[1:0] | Message Storage Indicator<br>00: No FIFO selected<br>01: FIFO message lost<br>10: Message stored in FIFO 0<br>11: Message stored in FIFO 1 |
| 5 to 0 | BIDX[5:0] | Buffer Index<br>Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = ‘1’. |

ISO CANFD ISO 11898-1:2015

### (24) MTTCAN0NDAT1 — New Data 1

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $098_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND31 | ND30 | ND29 | ND28 | ND27 | ND26 | ND25 | ND24 | ND23 | ND22 | ND21 | ND20 | ND19 | ND18 | ND17 | ND16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND15 | ND14 | ND13 | ND12 | ND11 | ND10 | ND9 | ND8 | ND7 | ND6 | ND5 | ND4 | ND3 | ND2 | ND1 | ND0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.96 MTTCAN0NDAT1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ND[31:0] | New Data<br>The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.<br>0: Rx Buffer not updated<br>1: Rx Buffer updated from new message |

ISO CANFD ISO 11898-1:2015

### (25) MTTCAN0NDAT2 — New Data 2

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $09C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND63 | ND62 | ND61 | ND60 | ND59 | ND58 | ND57 | ND56 | ND55 | ND54 | ND53 | ND52 | ND51 | ND50 | ND49 | ND48 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND47 | ND46 | ND45 | ND44 | ND43 | ND42 | ND41 | ND40 | ND39 | ND38 | ND37 | ND36 | ND35 | ND34 | ND33 | ND32 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.97 MTTCAN0NDAT2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ND[63:32] | New Data<br>The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.<br>0: Rx Buffer not updated<br>1: Rx Buffer updated from new message |

ISO CANFD ISO 11898-1:2015

### (26) MTTCAN0RXF0C — Rx FIFO 0 Configuration

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0A0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F0OM | F0WM[6:0] | | | | | | | — | F0S[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F0SA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.98 MTTCAN0RXF0C Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | F0OM | FIFO 0 Operation Mode<br>FIFO 0 can be operated in blocking or in overwrite mode (see **(2) Rx FIFOs**).<br>0: FIFO 0 blocking mode<br>1: FIFO 0 overwrite mode |
| 30 to 24 | F0WM[6:0] | Rx FIFO 0 Watermark<br>0: Watermark interrupt disabled<br>1 to 64: Level for Rx FIFO 0 watermark interrupt (MTTCAN0IR.RF0W)<br>>64: Watermark interrupt disabled |
| 23 | Reserved | This bit is always read as 0. |
| 22 to 16 | F0S[6:0] | Rx FIFO 0 Size<br>0: No Rx FIFO 0<br>1 to 64: Number of Rx FIFO 0 elements<br>>64: Values greater than 64 are interpreted as 64<br>The Rx FIFO 0 elements are indexed from 0 to F0S-1 |
| 15 to 2 | F0SA[15:2] | Rx FIFO 0 Start Address<br>Start address of Rx FIFO 0 in Message RAM (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. |

ISO CANFD (ISO 11898-1:2015

### (27) MTTCAN0RXF0S — Rx FIFO 0 Status

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0A4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RF0L | F0F | — | — | F0PI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | F0GI[5:0] | | | | | | — | F0FL[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.99 MTTCAN0RXF0S Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. |
| 25 | RF0L | Rx FIFO 0 Message Lost<br>This bit is a copy of interrupt flag MTTCAN0IR.RF0L. When MTTCAN0IR.RF0L is reset, this bit is also reset.<br>0: No Rx FIFO 0 message lost<br>1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero<br>**NOTE**<br>Overwriting the oldest message when MTTCAN0RXF0C.F0OM = '1' will not set this flag. |
| 24 | F0F | Rx FIFO 0 Full<br>0: Rx FIFO 0 not full<br>1: Rx FIFO 0 full |
| 23, 22 | Reserved | These bits are always read as 0. |
| 21 to 16 | F0PI[5:0] | Rx FIFO 0 Put Index<br>Rx FIFO 0 write index pointer, range 0 to 63. |
| 15, 14 | Reserved | These bits are always read as 0. |
| 13 to 8 | F0GI[5:0] | Rx FIFO 0 Get Index<br>Rx FIFO 0 read index pointer, range 0 to 63. |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 0 | F0FL[6:0] | Rx FIFO 0 Fill Level<br>Number of elements stored in Rx FIFO 0, range 0 to 64. |

ISO CANFD conformance 2015

### (28) MTTCAN0RXF0A — Rx FIFO 0 Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0A8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | F0AI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.100 MTTCAN0RXF0A Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 to 0 | F0AI[5:0] | Rx FIFO 0 Acknowledge Index<br>After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index MTTCAN0RXF0S.F0GI to F0AI + 1 and update the FIFO 0 Fill Level MTTCAN0RXF0S.F0FL. |

ISO CANFD ISO 11898-1:2015

### (29) MTTCAN0RXBC — Rx Buffer Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0AC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RBSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.101 MTTCAN0RXBC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 2 | RBSA[15:2] | Rx Buffer Start Address<br>Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C. |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

ISO CANFD ISO 11898-1:2015

### (30) MTTCAN0RXF1C — Rx FIFO 1 Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0B0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1OM | F1WM[6:0] | | | | | | | — | F1S[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1SA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.102 MTTCAN0RXF1C Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | F1OM | FIFO 1 Operation Mode<br>FIFO 1 can be operated in blocking or in overwrite mode (see **(2) Rx FIFOs**).<br>0: FIFO 1 blocking mode<br>1: FIFO 1 overwrite mode |
| 30 to 24 | F1WM[6:0] | Rx FIFO 1 Watermark<br>0: Watermark interrupt disabled<br>1 to 64: Level for Rx FIFO 1 watermark interrupt (MTTCAN0IR.RF1W)<br>>64: Watermark interrupt disabled |
| 23 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 22 to 16 | F1S[6:0] | Rx FIFO 1 Size<br>0: No Rx FIFO 1<br>1 to 64: Number of Rx FIFO 1 elements<br>>64: Values greater than 64 are interpreted as 64<br>The Rx FIFO 1 elements are indexed from 0 to F1S - 1 |
| 15 to 2 | F1SA[15:2] | Rx FIFO 1 Start Address<br>Start address of Rx FIFO 1 in Message RAM (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

ISO CANFD (ISO 11898-1:2015

### (31) MTTCAN0RXF1S — Rx FIFO 1 Status

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0B4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMS[1:0] | | — | — | — | — | RF1L | F1F | — | — | F1PI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | F1GI[5:0] | | | | | | — | F1FL[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.103 MTTCAN0RXF1S Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | DMS[1:0] | Debug Message Status<br>00: Idle state, wait for reception of debug messages, DMA request is cleared<br>01: Debug message A received<br>10: Debug messages A, B received<br>11: Debug messages A, B, C received, DMA request is set |
| 29 to 26 | Reserved | These bits are always read as 0. |
| 25 | RF1L | Rx FIFO 1 Message Lost<br>This bit is a copy of interrupt flag MTTCAN0IR.RF1L. When MTTCAN0IR.RF1L is reset, this bit is also reset.<br>0: No Rx FIFO 1 message lost<br>1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero<br>**NOTE**<br>Overwriting the oldest message when MTTCAN0RXF0C.F0OM = '1' will not set this flag. |
| 24 | F1F | Rx FIFO 1 Full<br>0: Rx FIFO 1 not full<br>1: Rx FIFO 1 full |
| 23, 22 | Reserved | These bits are always read as 0. |
| 21 to 16 | F1PI[5:0] | Rx FIFO 1 Put Index<br>Rx FIFO 1 write index pointer, range 0 to 63. |
| 15, 14 | Reserved | These bits are always read as 0. |
| 13 to 8 | F1GI[5:0] | Rx FIFO 1 Get Index<br>Rx FIFO 1 read index pointer, range 0 to 63. |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 0 | F1FL[6:0] | Rx FIFO 1 Fill Level<br>Number of elements stored in Rx FIFO 1, range 0 to 64. |

ISO CANFD IP Core - 2015

### (32) MTTCAN0RXF1A — Rx FIFO 1 Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0B8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | F1AI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.104 MTTCAN0RXF1A Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 to 0 | F1AI[5:0] | Rx FIFO 1 Acknowledge Index<br>After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index MTTCAN0RXF1S.F1GI to F1AI + 1 and update the FIFO 1 Fill Level MTTCAN0RXF1S.F1FL. |

ISO CANFD ISO 11898-1:2015

### (33) MTTCAN0RXESC — Rx Buffer / FIFO Element Size Configuration

Configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0BC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RBDS[2:0] | | | — | F1DS[2:0] | | | — | F0DS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | RP | RP | RP | R | RP | RP | RP | R | RP | RP | RP |

**Table 19.105 MTTCAN0RXESC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | These bits are always read as 0. |
| 10 to 8 | RBDS[2:0] | Rx Buffer Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 4 | F1DS[2:0] | Rx FIFO 1 Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |
| 3 | Reserved | This bit is always read as 0. |
| 2 to 0 | F0DS[2:0] | Rx FIFO 0 Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |

**NOTE**

In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by MTTCAN0RXESC are stored to the Rx Buffer resp. Rx FIFO element. The rest of the frame's data field is ignored.

ISO CANFD ISO 11898-1:2015

### (34) MTTCAN0TXBC — Tx Buffer Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0C0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TFQM | TFQS[5:0] | | | | | | — | — | NDTB[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | RP | RP | RP | RP | RP | RP | RP | R | R | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.106 MTTCAN0TXBC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 30 | TFQM | Tx FIFO/Queue Mode<br>0: Tx FIFO operation<br>1: Tx Queue operation |
| 29 to 24 | TFQS[5:0] | Transmit FIFO/Queue Size<br>0: No Tx FIFO/Queue<br>1 to 32: Number of Tx Buffers used for Tx FIFO/Queue<br>>32: Values greater than 32 are interpreted as 32 |
| 23, 22 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 21 to 16 | NDTB[5:0] | Number of Dedicated Transmit Buffers<br>0: No Dedicated Tx Buffers<br>1 to 32: Number of Dedicated Tx Buffers<br>>32: Values greater than 32 are interpreted as 32 |
| 15 to 2 | TBSA[15:2] | Tx Buffers Start Address<br>Start address of Tx Buffers section in Message RAM (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

**NOTE**

Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

ISO CANFD 11898-1:2015

### (35) MTTCAN0TXFQS — Tx FIFO/Queue Status

The Tx FIFO/Queue status is related to the pending Tx requests listed in register MTTCAN0TXBRP. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (MTTCAN0TXBRP not yet updated).

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0C4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | TFQF | TFQPI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | TFGI[4:0] | | | | | — | — | TFFL[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.107 MTTCAN0TXFQS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | These bits are always read as 0. |
| 21 | TFQF | Tx FIFO/Queue Full<br>0: Tx FIFO/Queue not full<br>1: Tx FIFO/Queue full |
| 20 to 16 | TFQPI[4:0] | Tx FIFO/Queue Put Index<br>Tx FIFO/Queue write index pointer, range 0 to 31. |
| 15 to 13 | Reserved | These bits are always read as 0. |
| 12 to 8 | TFGI[4:0] | Tx FIFO Get Index<br>Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (MTTCAN0TXBC.TFQM = '1'). |
| 7, 6 | Reserved | These bits are always read as 0. |
| 5 to 0 | TFFL[5:0] | Tx FIFO Free Level<br>Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (MTTCAN0TXBC.TFQM = '1') |

**NOTE**

In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers.
Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

ISO CANFD FDIS 11898-1:2015

### (36) MTTCAN0TXESC — Tx Buffer Element Size Configuration

Configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0C8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TBDS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP |

**Table 19.108 MTTCAN0TXESC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 to 0 | TBDS[2:0] | Tx Buffer Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |

**NOTE**

In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size MTTCAN0TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as "$CC_H$" (padding bytes).

ISO CANFD ISO 11898-1:2015

### (37) MTTCAN0TXBRP — Tx Buffer Request Pending

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0CC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRP31 | TRP30 | TRP29 | TRP28 | TRP27 | TRP26 | TRP25 | TRP24 | TRP23 | TRP22 | TRP21 | TRP20 | TRP19 | TRP18 | TRP17 | TRP16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRP15 | TRP14 | TRP13 | TRP12 | TRP11 | TRP10 | TRP9 | TRP8 | TRP7 | TRP6 | TRP5 | TRP4 | TRP3 | TRP2 | TRP1 | TRP0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.109 MTTCAN0TXBRP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TRP[31:0] | Transmission Request Pending<br>Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register MTTCAN0TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register MTTCAN0TXBCR. MTTCAN0TXBRP bits are set only for those Tx Buffers configured via MTTCAN0TXBC. After a MTTCAN0TXBRP bit has been set, a Tx scan (see **Section 19.5.3.5, Tx Handling**) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID).<br>A cancellation request resets the corresponding transmission request pending bit of register MTTCAN0TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding MTTCAN0TXBRP bit has been reset.<br>After a cancellation has been requested, a finished cancellation is signalled via MTTCAN0TXBCF<br>• after successful transmission together with the corresponding MTTCAN0TXBTO bit<br>• when the transmission has not yet been started at the point of cancellation<br>• when the transmission has been aborted due to lost arbitration<br>• when an error occurred during frame transmission<br>In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding MTTCAN0TXBCF bit is set for all unsuccessful transmissions.<br>0: No transmission request pending<br>1: Transmission request pending<br>**NOTE**<br>MTTCAN0TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding MTTCAN0TXBRP bit is reset. |

ISO CANFD IP Module © 2015

### (38) MTTCAN0TXBAR — Tx Buffer Add Request

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0D0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AR31 | AR30 | AR29 | AR28 | AR27 | AR26 | AR25 | AR24 | AR23 | AR22 | AR21 | AR20 | AR19 | AR18 | AR17 | AR16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AR15 | AR14 | AR13 | AR12 | AR11 | AR10 | AR9 | AR8 | AR7 | AR6 | AR5 | AR4 | AR3 | AR2 | AR1 | AR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.110 MTTCAN0TXBAR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | AR[31:0] | Add Request<br>Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to MTTCAN0TXBAR. MTTCAN0TXBAR bits are set only for those Tx Buffers configured via MTTCAN0TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.<br>0: No transmission request added<br>1: Transmission requested added |

**NOTE**

If an add request is applied for a Tx Buffer with pending transmission request (corresponding MTTCAN0TXBRP bit already set), this add request is ignored.

ISO CAN FD ISO 11898-1:2015

### (39) MTTCAN0TXBCR — Tx Buffer Cancellation Request

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0D4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CR31 | CR30 | CR29 | CR28 | CR27 | CR26 | CR25 | CR24 | CR23 | CR22 | CR21 | CR20 | CR19 | CR18 | CR17 | CR16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CR15 | CR14 | CR13 | CR12 | CR11 | CR10 | CR9 | CR8 | CR7 | CR6 | CR5 | CR4 | CR3 | CR2 | CR1 | CR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.111 MTTCAN0TXBCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CR[31:0] | Cancellation Request<br>Each Tx Buffer has its own Cancellation Request bit. Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to MTTCAN0TXBCR. MTTCAN0TXBCR bits are set only for those Tx Buffers configured via MTTCAN0TXBC. The bits remain set until the corresponding bit of MTTCAN0TXBRP is reset.<br>0: No cancellation pending<br>1: Cancellation pending |

ISO CANFD ISO 11898-1:2015

### (40) MTTCAN0TXBTO — Tx Buffer Transmission Occurred

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0D8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TO31 | TO30 | TO29 | TO28 | TO27 | TO26 | TO25 | TO24 | TO23 | TO22 | TO21 | TO20 | TO19 | TO18 | TO17 | TO16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TO15 | TO14 | TO13 | TO12 | TO11 | TO10 | TO9 | TO8 | TO7 | TO6 | TO5 | TO4 | TO3 | TO2 | TO1 | TO0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.112 MTTCAN0TXBTO Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TO[31:0] | Transmission Occurred<br>Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding MTTCAN0TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register MTTCAN0TXBAR.<br>0: No transmission occurred<br>1: Transmission occurred |

ISO CANFD ISO 11898-1:2015

### (41) MTTCAN0TXBCF — Tx Buffer Cancellation Finished

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0DC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF31 | CF30 | CF29 | CF28 | CF27 | CF26 | CF25 | CF24 | CF23 | CF22 | CF21 | CF20 | CF19 | CF18 | CF17 | CF16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF15 | CF14 | CF13 | CF12 | CF11 | CF10 | CF9 | CF8 | CF7 | CF6 | CF5 | CF4 | CF3 | CF2 | CF1 | CF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.113 MTTCAN0TXBCF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CF[31:0] | Cancellation Finished<br>Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding MTTCAN0TXBRP bit is cleared after a cancellation was requested via MTTCAN0TXBCR. In case the corresponding MTTCAN0TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register MTTCAN0TXBAR.<br>0: No transmit buffer cancellation<br>1: Transmit buffer cancellation finished |

### (42) MTTCAN0TXBTIE — Tx Buffer Transmission Interrupt Enable

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0E0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIE31 | TIE30 | TIE29 | TIE28 | TIE27 | TIE26 | TIE25 | TIE24 | TIE23 | TIE22 | TIE21 | TIE20 | TIE19 | TIE18 | TIE17 | TIE16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIE15 | TIE14 | TIE13 | TIE12 | TIE11 | TIE10 | TIE9 | TIE8 | TIE7 | TIE6 | TIE5 | TIE4 | TIE3 | TIE2 | TIE1 | TIE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.114 MTTCAN0TXBTIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TIE[31:0] | Transmission Interrupt Enable<br>Each Tx Buffer has its own Transmission Interrupt Enable bit.<br>0: Transmission interrupt disabled<br>1: Transmission interrupt enable |

ISO CANFD ISO 11898-1:2015

### (43) MTTCAN0TXBCIE — Tx Buffer Cancellation Finished Interrupt Enable

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0E4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIE31 | CFIE30 | CFIE29 | CFIE28 | CFIE27 | CFIE26 | CFIE25 | CFIE24 | CFIE23 | CFIE22 | CFIE21 | CFIE20 | CFIE19 | CFIE18 | CFIE17 | CFIE16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIE15 | CFIE14 | CFIE13 | CFIE12 | CFIE11 | CFIE10 | CFIE9 | CFIE8 | CFIE7 | CFIE6 | CFIE5 | CFIE4 | CFIE3 | CFIE2 | CFIE1 | CFIE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 19.115 MTTCAN0TXBCIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CFIE[31:0] | Cancellation Finished Interrupt Enable<br>Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit.<br>0: Cancellation finished interrupt disabled<br>1: Cancellation finished interrupt enabled |

ISO CANFD ISO 11898-1:2015

### (44) MTTCAN0TXEFC — Tx Event FIFO Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0F0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | EFWM[5:0] | | | | | | — | — | EFS[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | RP | RP | RP | RP | RP | RP | R | R | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EFSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.116 MTTCAN0TXEFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 29 to 24 | EFWM[5:0] | Event FIFO Watermark<br>0: Watermark interrupt disabled<br>1 to 32: Level for Tx Event FIFO watermark interrupt (MTTCAN0IR.TEFW)<br>>32: Watermark interrupt disabled |
| 23, 22 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 21 to 16 | EFS[5:0] | Event FIFO Size<br>0: Tx Event FIFO disabled<br>1 to 32: Number of Tx Event FIFO elements<br>>32: Values greater than 32 are interpreted as 32<br>The Tx Event FIFO elements are indexed from 0 to EFS – 1 |
| 15 to 2 | EFSA[15:2] | Event FIFO Start Address<br>Start address of Tx Event FIFO in Message RAM (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

ISO CANFD CO Mode - 2015

### (45) MTTCAN0TXEFS — Tx Event FIFO Status

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0F4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TEFL | EFF | — | — | — | EFPI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | EFGI[4:0] | | | | | — | — | EFFL[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.117 MTTCAN0TXEFS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. |
| 25 | TEFL | Tx Event FIFO Element Lost<br>This bit is a copy of interrupt flag MTTCAN0IR.TEFL. When MTTCAN0IR.TEFL is reset, this bit is also reset.<br>0: No Tx Event FIFO element lost<br>1: Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero. |
| 24 | EFF | Event FIFO Full<br>0: Tx Event FIFO not full<br>1: Tx Event FIFO full |
| 23 to 21 | Reserved | These bits are always read as 0. |
| 20 to 16 | EFPI[4:0] | Event FIFO Put Index<br>Tx Event FIFO write index pointer, range 0 to 31. |
| 15 to 13 | Reserved | These bits are always read as 0. |
| 12 to 8 | EFGI[4:0] | Event FIFO Get Index<br>Tx Event FIFO read index pointer, range 0 to 31. |
| 7, 6 | Reserved | These bits are always read as 0. |
| 5 to 0 | EFFL[5:0] | Event FIFO Fill Level<br>Number of elements stored in Tx Event FIFO, range 0 to 32. |

ISO CANFD ISO 11898-1:2015

### (46) MTTCAN0TXEFA — Tx Event FIFO Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0F8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | EFAI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 19.118 MTTCAN0TXEFA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 4 to 0 | EFAI[4:0] | Event FIFO Acknowledge Index<br>After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index MTTCAN0TXEFS.EFGI to EFAI + 1 and update the FIFO 0 Fill Level MTTCAN0TXEFS.EFFL. |

ISO CANFD ISO 11898-1:2015

### (47) MTTCAN0TTTMC — TT Trigger Memory Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $100_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | TME[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 19.119 MTTCAN0TTTMC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 22 to 16 | TME[6:0] | Trigger Memory Elements<br>0: No Trigger Memory<br>1 to 64: Number of Trigger Memory elements<br>>64: Values greater than 64 are interpreted as 64 |
| 15 to 2 | TMSA[15:2] | Trigger Memory Start Address<br>Start address of Trigger Memory in Message RAM (32-bit word address, see **Figure 19.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

ISO CANFD ISO 11898-1:2015

### (48) MTTCAN0TTRMC — TT Reference Message Configuration

For details about handling of reference messages see **(1) Reference Message**

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $104_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMPS | XTD | — | RID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

**Table 19.120 MTTCAN0TTRMC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | RMPS | Reference Message Payload Select<br>Ignored in case of time slaves.<br>0: Reference message has no additional payload<br>1: The following elements are taken from Tx Buffer 0:<br>Message Marker MM, Event FIFO Control EFC, Data Length Code DLC, Data Bytes DB<br>(Level 1: bytes 2 to 8, Level 0,2: bytes 5 to 8) |
| 30 | XTD | Extended Identifier<br>0: 11-bit standard identifier<br>1: 29-bit extended identifier |
| 29 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 28 to 0 | RID[28:0] | Reference Identifier<br>Identifier transmitted with reference message and used for reference message filtering. Standard or extended reference identifier depending on bit XTD. A standard identifier has to be written to ID[28:18]. |

ISO CANFD ISP Rev.1.1:2015

### (49) MTTCAN0TTOCF — TT Operation Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $108_H$

**Value after reset:** $0001\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | EVTP | ECC | EGTF | AWL[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EECS | IRTO[6:0] | | | | | | | LDSDL[2:0] | | | TM | GEN | — | OM[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP |

**Table 19.121 MTTCAN0TTOCF Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 26 | EVTP | Event Trigger Polarity<br>0: Rising edge trigger<br>1: Falling edge trigger |
| 25 | ECC | Enable Clock Calibration<br>0: Automatic clock calibration in TTCAN Level 0, 2 is disabled<br>1: Automatic clock calibration in TTCAN Level 0, 2 is enabled |
| 24 | EGTF | Enable Global Time Filtering<br>0: Global time filtering in TTCAN Level 0, 2 is disabled<br>1: Global time filtering in TTCAN Level 0, 2 is enabled |
| 23 to 16 | AWL[7:0] | Application Watchdog Limit<br>The application watchdog can be disabled by programming AWL to 00H.<br>$00_H$ to $FF_H$: Maximum time after which the application has to serve the application watchdog.<br>The application watchdog is incremented once each 256 NTUs. |
| 15 | EECS | Enable External Clock Synchronization<br>If enabled, TUR configuration (MTTCAN0TURCF.NCL only) may be updated during TTCAN operation.<br>0: External clock synchronization in TTCAN Level 0, 2 disabled<br>1: External clock synchronization in TTCAN Level 0, 2 enabled |
| 14 to 8 | IRTO[6:0] | Initial Reference Trigger Offset<br>$00_H$ to $7F_H$: Positive offset, range from 0 to 127 |
| 7 to 5 | LDSDL[2:0] | LD of Synchronization Deviation Limit<br>The Synchronization Deviation Limit SDL is configured by its dual logarithm LDSDL with SDL = $2^{(LDSDL + 5)}$. It should not exceed the clock tolerance given by the CAN bit timing configuration.<br>$0_H$ to $7_H$: LD of Synchronization Deviation Limit (SDL ≤ 32 to 4096) |
| 4 | TM | Time Master<br>0: Time Master function disabled<br>1: Potential Time Master |

ISO CANFD (ISO 11898-1:2015

**Table 19.121 MTTCAN0TTOCF Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 | GEN | Gap Enable<br>0: Strictly time-triggered operation<br>1: External event-synchronized time-triggered operation |
| 2 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 1, 0 | OM[1:0] | Operation Mode<br>00: Event-driven CAN communication, default<br>01: TTCAN level 1<br>10: TTCAN level 2<br>11: TTCAN level 0 |

ISO CANFD ISO 11898-1:2015

### (50) MTTCAN0TTMLM — TT Matrix Limits

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $10C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ENTT[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TXEW[3:0] | | | | CSS[1:0] | | CCM[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

**Table 19.122 MTTCAN0TTMLM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 27 to 16 | ENTT[11:0] | Expected Number of Tx Triggers<br>$000_H$ to $FFF_H$: Expected number of Tx Triggers in one Matrix Cycle |
| 15 to 12 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 11 to 8 | TXEW[3:0] | Tx Enable Window<br>$0_H$ to $F_H$: Length of Tx enable window, 1 to 16 NTU cycles |
| 7, 6 | CSS[1:0] | Cycle Start Synchronization<br>Enables sync pulse output at pin m_ttcan_soc (MTTCAN0SOC).<br>00: No sync pulse<br>01: Sync pulse at start of basic cycle<br>10: Sync pulse at start of matrix cycle<br>11: Reserved |
| 5 to 0 | CCM[5:0] | Cycle Count Max<br>$00_H$: 1 Basic Cycle per Matrix Cycle<br>$01_H$: 2 Basic Cycles per Matrix Cycle<br>$03_H$: 4 Basic Cycles per Matrix Cycle<br>$07_H$: 8 Basic Cycles per Matrix Cycle<br>$0F_H$: 16 Basic Cycles per Matrix Cycle<br>$1F_H$: 32 Basic Cycles per Matrix Cycle<br>$3F_H$: 64 Basic Cycles per Matrix Cycle<br>others: Reserved |

**NOTE**

*ISO 11898-4, Section 5.2.1 requires,* that only the listed cycle count values are configured. Other values are possible but may lead to inconsistent matrix cycles.

ISO CANFD ISO 11898-1:2015

### (51) MTTCAN0TURCF — TUR Configuration

The length of the NTU is given by: NTU = CAN Clock Period • NC/DC

NC is an 18-bit value. Its high part, NCH[17:16] is hard wired to 0b01. Therefore the range of NC is $10000_H$ to $1FFFF_H$. The value configured by NCL is the initial value for MTTCAN0TURNA.NAV[15:0]. DC is set to $1000_H$ by hardware reset and it may not be written to $0000_H$.

Level 1: NC ≥ 4 • DC and NTU = CAN bit time
Level 0, 2: NC ≥ 8 • DC

The actual value of TUR may be changed by the clock drift compensation function of TTCAN Level 0 and Level 2 in order to adjust the node's local view of the NTU to the time master's view of the NTU. DC will not be changed by the automatic drift compensation, MTTCAN0TURNA.NAV may be adjusted around NC in the range of the Synchronization Deviation Limit given by MTTCAN0TTOCF.LDSDL. NC and DC should be programmed to the largest suitable values in order to allow the best computational accuracy for the drift compensation process.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $110_H$

**Value after reset:** $1000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ELT | — | DC[13:0] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NCL[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

ISO CANFD ISO 11898-1:2015

**Table 19.123 MTTCAN0TURCF Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | ELT | Enable Local Time<br>0: Local time is stopped, default<br>1: Local time is enabled<br>**NOTE**<br>Local time is started by setting ELT. It remains active until ELT is reset or until the next hardware reset. MTTCAN0TURCF.DC is locked when MTTCAN0TURCF.ELT = '1'. If ELT is written to '0', the readable value will stay at '1' until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register remains locked. |
| 30 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 29 to 16 | DC[13:0] | Denominator Configuration<br>$0000_H$: Illegal value<br>$0001_H$ to $3FFF_H$: Denominator Configuration |

**Table 19.123 MTTCAN0TURCF Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | NCL[15:0] | Numerator Configuration Low<br>Write access to the TUR Numerator Configuration Low is only possible during configuration with MTTCAN0TURCF.ELT = '0' or if MTTCAN0TTOCF.EECS (external clock synchronization enabled) is set. When a new value for NCL is written outside TT Configuration Mode, the new value takes effect when MTTCAN0TTOST.WECS is cleared to '0'. NCL is locked MTTCAN0TTOST.WECS is '1'.<br>$0000_H$ to $FFFF_H$: Numerator Configuration Low<br>**NOTE**<br>If NC < 7 • DC in TTCAN Level 1, then it is required that subsequent time marks in the Trigger Memory must differ by at least 2 NTU. |

ISO CANFD ISO 11898-1:2015

### (52) MTTCAN0TTOCN — TT Operation Control

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $114_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LCKC | — | ESCN | NIG | TMG | FGP | GCS | TTIE | TMC[1:0] | | RTIE | SWS[1:0] | | SWP | ECS | SGT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW |

**Table 19.124 MTTCAN0TTOCN Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 | LCKC | TT Operation Control Register Locked<br>Set by a write access to register MTTCAN0TTOCN. Reset when the updated configuration has been synchronized into the CAN clock domain.<br>0: Write access to MTTCAN0TTOCN enabled<br>1: Write access to MTTCAN0TTOCN locked |
| 14 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 13 | ESCN | External Synchronization Control<br>If enabled the M_TTCAN synchronizes its cycle time phase to an external event signalled by a rising edge at pin m_ttcan_evt (MTTCAN0EVT)(see **Section 19.6.4.11, Synchronization to external Time Schedule**).<br>0: External synchronization disabled<br>1: External synchronization enabled |
| 12 | NIG | Next is Gap<br>This bit can only be set when the M_TTCAN is the actual Time Master and when it is configured for external event-synchronized time-triggered operation (MTTCAN0TTOCF.GEN = '1')<br>0: No action, reset by reception of any reference message<br>1: Transmit next reference message with Next_is_Gap = '1' |
| 11 | TMG | Time Mark Gap<br>0: Reset by each reference message<br>1: Next reference message started when Register Time Mark interrupt MTTCAN0TTIR.RTMI is activated |
| 10 | FGP | Finish Gap<br>Set by the CPU, reset by each reference message<br>0: No reference message requested<br>1: Application requested start of reference message |
| 9 | GCS | Gap Control Select<br>0: Gap control independent from m_ttcan_evt (MTTCAN0EVT)<br>1: Gap control by input pin m_ttcan_evt (MTTCAN0EVT) |
| 8 | TTIE | Trigger Time Mark Interrupt Pulse Enable<br>External time mark events are configured by trigger memory element TMEX (see **(7) Trigger Memory Element**). A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the M_TTCAN is in synchronization state In_Schedule or In_Gap.<br>0: Trigger Time Mark Interrupt output m_ttcan_tmp (MTTCAN0TMP) disabled<br>1: Trigger Time Mark Interrupt output m_ttcan_tmp (MTTCAN0TMP) enabled |

ISO CANFD IP Module - 2015

**Table 19.124 MTTCAN0TTOCN Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | TMC[1:0] | Register Time Mark Compare<br>00: No Register Time Mark Interrupt generated<br>01: Register Time Mark Interrupt if Time Mark = cycle time<br>10: Register Time Mark Interrupt if Time Mark = local time<br>11: Register Time Mark Interrupt if Time Mark = global time<br>**NOTE**<br>When changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = “00”, then reconfigure MTTCAN0TTTMK, and finally set TMC to the intended time reference. |
| 5 | RTIE | Register Time Mark Interrupt Pulse Enable<br>Register time mark interrupts are configured by register MTTCAN0TTTMK. A register time mark interrupt pulse with the length of one m_ttcan_clk period is generated when the time referenced by MTTCAN0TTOCN.TMC (cycle, local, or global) equals MTTCAN0TTTMK.TM, independent of the synchronization state.<br>0: Register Time Mark Interrupt output m_ttcan_rtp (MTTCAN0RTP) disabled<br>1: Register Time Mark Interrupt output m_ttcan_rtp (MTTCAN0RTP) enabled |
| 4, 3 | SWS[1:0] | Stop Watch Source<br>00: Stop Watch disabled<br>01: Actual value of cycle time is copied to MTTCAN0TTCPT.SWV<br>10: Actual value of local time is copied to MTTCAN0TTCPT.SWV<br>11: Actual value of global time is copied to MTTCAN0TTCPT.SWV |
| 2 | SWP | Stop Watch Polarity<br>0: Rising edge trigger<br>1: Falling edge trigger |
| 1 | ECS | External Clock Synchronization<br>Writing a ‘1’ to ECS sets MTTCAN0TTOST.WECS if the node is the actual Time Master. ECS is reset after one Host clock period. The external clock synchronization takes effect at the start of the next basic cycle. |
| 0 | SGT | Set Global time<br>Writing a ‘1’ to SGT sets MTTCAN0TTOST.WGDT if the node is the actual Time Master. SGT is reset after one Host clock period. The global time preset takes effect when the node transmits the next reference message with the Master_Ref_Mark modified by the preset value written to MTTCAN0TTGTP. |

ISO CANFD IC V1.0 Dec. 2015

### (53) MTTCAN0TTGTP — TT Global Time Preset

If MTTCAN0TTOST.WGDT is set, the next reference message will be transmitted with the Master_Ref_Mark modified by the preset value and with Disc_Bit = '1', presetting the global time in all nodes simultaneously.

TP is reset to $0000_H$ each time a reference message with Disc_Bit = '1' becomes valid or if the node is not the current Time Master. TP is locked while MTTCAN0TTOST.WGTD = '1' after setting MTTCAN0TTOCN.SGT until the reference message with Disc_Bit = '1' becomes valid or until the node is no longer the current Time Master.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $118_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CTP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

ISO CAN FD ISO 11898-1:2015

**Table 19.125 MTTCAN0TTGTP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | CTP[15:0] | Cycle Time Target Phase<br>CTP is write-protected while MTTCAN0TTOCN.ESCN or MTTCAN0TTOST.SPL are set (see **Section 19.6.4.11, Synchronization to external Time Schedule**).<br>$0000_H$ to $FFFF_H$: Defines target value of cycle time when a rising edge of m_ttcan_evt (MTTCAN0EVT) is expected |
| 15 to 0 | TP[15:0] | Time Preset<br>TP is write-protected while MTTCAN0TTOST.WGTD is set.<br>$0000_H$ to $7FFF_H$: Next Master Reference Mark = Master Reference Mark + TP<br>$8000_H$: reserved<br>$8001_H$ to $FFFF_H$: Next Master Reference Mark = Master Reference Mark – ($10000_H$ – TP) |

### (54) MTTCAN0TTTMK — TT Time Mark

A time mark interrupt (MTTCAN0TTIR.RTMI = '1') is generated when the time base indicated by MTTCAN0TTOCN.TMC (cycle time, local time, or global time) has the same value as TM.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $11C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LCKM | — | — | — | — | — | — | — | — | TICC[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | RW | R/W | RW | R/W | RW | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

**Table 19.126 MTTCAN0TTTMK Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | LCKM | TT Time Mark Register Locked<br>Always set by a write access to registers MTTCAN0TTOCN. Set by write access to register MTTCAN0TTTMK when MTTCAN0TTOCN.TMC ≠ "00". Reset when the registers have been synchronized into the CAN clock domain.<br>0: Write access to MTTCAN0TTTMK enabled<br>1: Write access to MTTCAN0TTTMK locked |
| 30 to 23 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 22 to 16 | TICC[6:0] | Time Mark Cycle Code<br>Cycle count for which the time mark is valid.<br>000000x: valid for all cycles<br>000001c: valid every second cycle at cycle count mod2 = c<br>00001cc: valid every fourth cycle at cycle count mod4 = cc<br>0001ccc: valid every eighth cycle at cycle count mod8 = ccc<br>001cccc: valid every sixteenth cycle at cycle count mod16 = cccc<br>01ccccc: valid every thirty-second cycle at cycle count mod32 = ccccc<br>1cccccc: valid every sixty-fourth cycle at cycle count mod64 = cccccc |
| 15 to 0 | TM[15:0] | Time Mark<br>$0000_H$ to $FFFF_H$: Time Mark |

**NOTE**

When using byte access to register MTTCAN0TTTMK it is recommended to first disable the time mark compare function (MTTCAN0TTOCN.TMC = "00") to avoid compares on inconsistent register values.

ISO CANFD 11898-1:2015

### (55) MTTCAN0TTIR — TT Interrupt Register

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $120_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CER | AW | WT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | RW | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IWT | ELC | SE2 | SE1 | TXO | TXU | GTE | GTD | GTW | SWE | TTMI | RTMI | SOG | CSM | SMC | SBC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

**Table 19.127 MTTCAN0TTIR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 18 | CER | Configuration Error<br>Trigger out of order.<br>0: No error found in trigger list<br>1: Error found in trigger list |
| 17 | AW | Application Watchdog<br>0: Application watchdog served in time<br>1: Application watchdog not served in time |
| 16 | WT | Watch Trigger<br>0: No missing reference message<br>1: Missing reference message (Level 0: cycle time FF00H) |
| 15 | IWT | Initialization Watch Trigger<br>The initialization is restarted by resetting IWT.<br>0: No missing reference message during system startup<br>1: No system startup due to missing reference message |
| 14 | ELC | Error Level Changed<br>Not set when error level changed during initialization.<br>0: No change in error level<br>1: Error level changed |
| 13 | SE2 | Scheduling Error 2<br>0: No scheduling error 2<br>1: Scheduling error 2 occurred |
| 12 | SE1 | Scheduling Error 1<br>0: No scheduling error 1<br>1: Scheduling error 1 occurred |
| 11 | TXO | Tx Count Overflow<br>0: Number of Tx Trigger as expected<br>1: More Tx trigger than expected in one matrix cycle |
| 10 | TXU | Tx Count Underflow<br>0: Number of Tx Trigger as expected<br>1: Less Tx trigger than expected in one matrix cycle |

ISO CANFD CC 11898-1:2015

**Table 19.127 MTTCAN0TTIR Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 9 | GTE | Global Time Error<br>Synchronization deviation SD exceeds limit specified by MTTCAN0TTOCF.LDSDL, TTCAN Level 0, 2 only.<br>0: Synchronization deviation within limit<br>1: Synchronization deviation exceeded limit |
| 8 | GTD | Global Time Discontinuity<br>0: No discontinuity of global time<br>1: Discontinuity of global time |
| 7 | GTW | Global Time Wrap<br>0: No global time wrap occurred<br>1: Global time wrap from $FFFF_H$ to $0000_H$ occurred |
| 6 | SWE | Stop Watch Event<br>0: No rising/falling edge at stop watch trigger pin m_ttcan_swt (MTTCAN0SWT) detected<br>1: Rising/falling edge at stop watch trigger pin m_ttcan_swt (MTTCAN0SWT) detected |
| 5 | TTMI | Trigger Time Mark Event Internal<br>Internal time mark events are configured by trigger memory element TMIN (see **(7) Trigger Memory Element**). Set when the trigger memory element becomes active, and the M_TTCAN is in synchronization state In_Gap or In_Schedule.<br>0: Time mark not reached<br>1: Time mark reached (Level 0: cycle time MTTCAN0TTOCF.IRTO • $200_H$) |
| 4 | RTMI | Register Time Mark Interrupt<br>Set when time referenced by MTTCAN0TTOCN.TMC (cycle, local, or global) equals MTTCAN0TTTMK.TM, independent of the synchronization state.<br>0: Time mark not reached<br>1: Time mark reached |
| 3 | SOG | Start of Gap<br>0: No reference message seen with Next_is_Gap bit set<br>1: Reference message with Next_is_Gap bit set becomes valid |
| 2 | CSM | Change of Synchronization Mode<br>0: No change in master to slave relation or schedule synchronization<br>1: Master to slave relation or schedule synchronization changed, also set when MTTCAN0TTOST.SPL is reset |
| 1 | SMC | Start of Matrix Cycle<br>0: No Matrix Cycle started since bit has been reset<br>1: Matrix Cycle started |
| 0 | SBC | Start of Basic Cycle<br>0: No Basic Cycle started since bit has been reset<br>1: Basic Cycle started |

ISO CANFD CC (ISO 11898-1:2015

### (56) MTTCAN0TTIE — TT Interrupt Enable

The settings in the TT Interrupt Enable register determine which status changes in the TT Interrupt Register will result in an interrupt.

0: TT interrupt disabled

1: TT interrupt enabled

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $124_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CERE | AWE | WTE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | RW | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IWTE | ELCE | SE2E | SE1E | TXOE | TXUE | GTEE | GTDE | GTWE | SWEE | TTMIE | RTMIE | SOGE | CSME | SMCE | SBCE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

ISO/CANFD (ISO 11898-1:2015

**Table 19.128 MTTCAN0TTIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 18 | CERE | Configuration Error Interrupt Enable |
| 17 | AWE | Application Watchdog Interrupt Enable |
| 16 | WTE | Watch Trigger Interrupt Enable |
| 15 | IWTE | Initialization Watch Trigger Interrupt Enable |
| 14 | ELCE | Change Error Level Interrupt Enable |
| 13 | SE2E | Scheduling Error 2 Interrupt Enable |
| 12 | SE1E | Scheduling Error 1 Interrupt Enable |
| 11 | TXOE | Tx Count Overflow Interrupt Enable |
| 10 | TXUE | Tx Count Underflow Interrupt Enable |
| 9 | GTEE | Global Time Error Interrupt Enable |
| 8 | GTDE | Global Time Discontinuity Interrupt Enable |
| 7 | GTWE | Global Time Wrap Interrupt Enable |
| 6 | SWEE | Stop Watch Event Interrupt Enable |
| 5 | TTMIE | Trigger Time Mark Event Internal Enable |
| 4 | RTMIE | Register Time Mark Interrupt Enable |
| 3 | SOGE | Start of Gap Interrupt Enable |
| 2 | CSME | Change of Synchronization Mode Interrupt Enable |
| 1 | SMCE | Start of Matrix Cycle Interrupt Enable |
| 0 | SBCE | Start of Basic Cycle Interrupt Enable |

### (57) MTTCAN0TTILS — TT Interrupt Line Select

The TT Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the TT Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via MTTCAN0ILE.EINT0 and MTTCAN0ILE.EINT1.

0: TT interrupt assigned to interrupt line m_ttcan_int0 (INTMTTCANI0)

1: TT interrupt assigned to interrupt line m_ttcan_int1 (INTMTTCANI1)

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $128_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CERL | AWL | WTL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | RW | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IWTL | ELCL | SE2L | SE1L | TXOL | TXUL | GTEL | GTDL | GTWL | SWEL | TTMIL | RTMIL | SOGL | CSML | SMCL | SBCL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

ISO CANFD FD 11898-1:2015

**Table 19.129 MTTCAN0TTILS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 18 | CERL | Configuration Error Interrupt Line |
| 17 | AWL | Application Watchdog Interrupt Line |
| 16 | WTL | Watch Trigger Interrupt Line |
| 15 | IWTL | Initialization Watch Trigger Interrupt Line |
| 14 | ELCL | Change Error Level Interrupt Line |
| 13 | SE2L | Scheduling Error 2 Interrupt Line |
| 12 | SE1L | Scheduling Error 1 Interrupt Line |
| 11 | TXOL | Tx Count Overflow Interrupt Line |
| 10 | TXUL | Tx Count Underflow Interrupt Line |
| 9 | GTEL | Global Time Error Interrupt Line |
| 8 | GTDL | Global Time Discontinuity Interrupt Line |
| 7 | GTWL | Global Time Wrap Interrupt Line |
| 6 | SWEL | Stop Watch Event Interrupt Line |
| 5 | TTMIL | Trigger Time Mark Event Internal Line |
| 4 | RTMIL | Register Time Mark Interrupt Line |
| 3 | SOGL | Start of Gap Interrupt Line |
| 2 | CSML | Change of Synchronization Mode Interrupt Line |
| 1 | SMCL | Start of Matrix Cycle Interrupt Line |
| 0 | SBCL | Start of Basic Cycle Interrupt Line |

### (58) MTTCAN0TTOST — TT Operation Status

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $12C_H$

**Value after reset:** $0000\ 0080_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPL | WECS | AWE | WFE | GSI | TMP[2:0] | | | GFI | WGTD | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RTO[7:0] | | | | | | | | QCS | QGTP | SYS[1:0] | | MS[1:0] | | EL[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.130 MTTCAN0TTOST Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | SPL | Schedule Phase Lock<br>The bit is valid only when external synchronization is enabled (MTTCAN0TTOCN.ESCN = '1'). In this case it signals that the difference between cycle time configured by MTTCAN0TTGTP.CTP and the cycle time at the rising edge at pin m_ttcan_evt (MTTCAN0EVT) is less or equal 9 NTU (see **Section 19.6.4.11, Synchronization to external Time Schedule**).<br>0: Phase outside range<br>1: Phase inside range |
| 30 | WECS | Wait for External Clock Synchronization<br>0: No external clock synchronization pending<br>1: Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle. |
| 29 | AWE | Application Watchdog Event<br>The application watchdog is served by reading MTTCAN0TTOST. When the watchdog is not served in time, bit AWE is set, all TTCAN communication is stopped, and the M_TTCAN is set into Bus Monitoring Mode.<br>0: Application Watchdog served in time<br>1: Failed to serve Application Watchdog in time |
| 28 | WFE | Wait for Event<br>0: No Gap announced, reset by a reference message with Next_is_Gap = '0'<br>1: Reference message with Next_is_Gap = '1' received |
| 27 | GSI | Gap Started Indicator<br>0: No Gap in schedule, reset by each reference message and for all time slaves<br>1: Gap time after Basic Cycle has started |
| 26 to 24 | TMP[2:0] | Time Master Priority<br>$0_H$ to $7_H$: Priority of actual Time Master |
| 23 | GFI | Gap Finished Indicator<br>Set when the CPU writes MTTCAN0TTOCN.FGP, or by a time mark interrupt if TMG = '1', or via input pin m_ttcan_evt (MTTCAN0EVT) if MTTCAN0TTOCN.GCS = '1'. Not set by Ref_Trigger_Gap or when Gap is finished by another node sending a reference message.<br>0: Reset at the end of each reference message<br>1: Gap finished by M_TTCAN |
| 22 | WGTD | Wait for Global Time Discontinuity<br>0: No global time preset pending<br>1: Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a reference message with Disc_Bit = '1' or after it received a reference message. |

ISO CANFD (ISO 11898-1:2015

**Table 19.130 MTTCAN0TTOST Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 21 to 16 | Reserved | These bits are always read as 0. |
| 15 to 8 | RTO[7:0] | Reference Trigger Offset<br>The Reference Trigger Offset value is a signed integer with a range from -127 ($81_H$) to 127 ($7F_H$). There is no notification when the lower limit of -127 is reached. In case the M_TTCAN becomes Time Master (MS[1:0] = “11”), the reset of RTO is delayed due to synchronization between Host and CAN clock domain. For time slaves the value configured by MTTCAN0TTOCF.IRTO is read.<br>$00_H$ to $FF_H$: Actual Reference Trigger offset value |
| 7 | QCS | Quality of Clock Speed<br>Only relevant in TTCAN Level 0 and Level 2, otherwise fixed to ‘1’.<br>0: Local clock speed not synchronized to Time Master clock speed<br>1: Synchronization Deviation ≤ SDL |
| 6 | QGTP | Quality of Global Time Phase<br>Only relevant in TTCAN Level 0 and Level 2, otherwise fixed to ‘0’.<br>0: Global time not valid<br>1: Global time in phase with Time Master |
| 5, 4 | SYS[1:0] | Synchronization State<br>00: Out of Synchronization<br>01: Synchronizing to TTCAN communication<br>10: Schedule suspended by Gap (In_Gap)<br>11: Synchronized to schedule (In_Schedule) |
| 3, 2 | MS[1:0] | Master State<br>00: Master_Off, no master properties relevant<br>01: Operating as Time Slave<br>10: Operating as Backup Time Master<br>11: Operating as current Time Master |
| 1, 0 | EL[1:0] | Error Level<br>00: Severity 0 - No Error<br>01: Severity 1 - Warning<br>10: Severity 2 - Error<br>11: Severity 3 - Severe Error |

ISO CANFD CC 11898-1:2015

### (59) MTTCAN0TURNA — TUR Numerator Actual

There is no drift compensation in TTCAN Level 1 (NAV = NC). In TTCAN Level 0 and Level 2, the drift between the node's local clock and the time master's local clock is calculated. The drift is compensated when the Synchronization Deviation (difference between NC and the calculated NAV) is not more than 2 (MTTCAN0TTOCF.LDSDL + 5). With MTTCAN0TTOCF.LDSDL ≤ 7, this results in a maximum range for NAV of $(NC - 1000_H) \leq NAV \leq (NC + 1000_H)$.

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $130_H$

**Value after reset:** $0001\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | NAV[17:16] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NAV[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.131 MTTCAN0TURNA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | These bits are always read as 0. |
| 17 to 0 | NAV[17:0] | Numerator Actual Value<br>≤ $0EFFF_H$: Illegal value<br>$0F000_H$ to $20FFF_H$: Actual numerator value<br>≥ $21000_H$: Illegal value |

ISO CANFD CC 11898-1:2015

### (60) MTTCAN0TTLGT — TT Local & Global Time

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $134_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.132 MTTCAN0TTLGT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | GT[15:0] | Global Time<br>Non-fractional part of the sum of the node's local time and its local offset (see **Section 19.6.4.5, Local Time, Cycle Time, Global Time, and External Clock Synchronization**).<br>$0000_H$ to $FFFF_H$: Global time value of TTCAN network |
| 15 to 0 | LT[15:0] | Local Time<br>Non-fractional part of local time, incremented once each local NTU (see **Section 19.6.4.5, Local Time, Cycle Time, Global Time, and External Clock Synchronization**).<br>$0000_H$ to $FFFF_H$: Local time value of TTCAN node |

ISO CANFD ISO 11898-1:2015

### (61) MTTCAN0TTCTC — TT Cycle Time & Count

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $138_H$

**Value after reset:** $003F\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CC[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.133 MTTCAN0TTCTC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | These bits are always read as 0. |
| 21 to 16 | CC[5:0] | Cycle Count<br>$00_H$ to $3F_H$: Number of actual Basic Cycle in the System Matrix |
| 15 to 0 | CT[15:0] | Cycle Time<br>Non-fractional part of the difference of the node's local time and Ref_Mark (see **Section 19.6.4.5, Local Time, Cycle Time, Global Time, and External Clock Synchronization**).<br>$0000_H$ to $FFFF_H$: Cycle time value of TTCAN Basic Cycle |

ISO CANFD ISO 11898-1:2015

### (62) MTTCAN0TTCPT — TT Capture Time

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $13C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SWV[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CCV[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.134 MTTCAN0TTCPT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | SWV[15:0] | Stop Watch Value<br>On a rising/falling edge (as configured via MTTCAN0TTOCN.SWP) at the Stop Watch Trigger pin m_ttcan_swt (MTTCAN0SWT) , when MTTCAN0TTOCN.SWS is ≠ “00” and MTTCAN0TTIR.SWE is ‘0’, the actual time value as selected by MTTCAN0TTOCN.SWS (cycle, local, global) is copied to SWV and MTTCAN0TTIR.SWE will be set to ‘1’. Capturing of the next stop watch value is enabled by resetting MTTCAN0TTIR.SWE.<br>$0000_H$ to $FFFF_H$: Captured Stop Watch value |
| 15 to 6 | Reserved | These bits are always read as 0. |
| 5 to 0 | CCV[5:0] | Cycle Count Value<br>Cycle count value captured together with SWV.<br>$00_H$ to $3F_H$: Captured cycle count value |

ISO CANFD [60 ... 8- ... 2015

### (63) MTTCAN0TTCSM — TT Cycle Sync Mark

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $140_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 19.135 MTTCAN0TTCSM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. |
| 15 to 0 | CSM[15:0] | Cycle Sync Mark<br>The Cycle Sync Mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid.<br>$0000_H$ to $FFFF_H$: Captured cycle time |

ISO CANFD ISO 11898-1:2015

### 19.6.2.4 Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration a single-ported Message RAM is connected to the M_TTCAN module.

#### (1) Message RAM Configuration

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_TTCAN module can be configured to allocate up to 4480 words in the Message RAM. It is not necessary to configure each of the sections listed in **Figure 19.16, Message RAM Configuration**, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via MTTCAN0RXESC.F0DS, MTTCAN0RXESC.F1DS, MTTCAN0RXESC.RBDS, and MTTCAN0TXESC.TBDS.

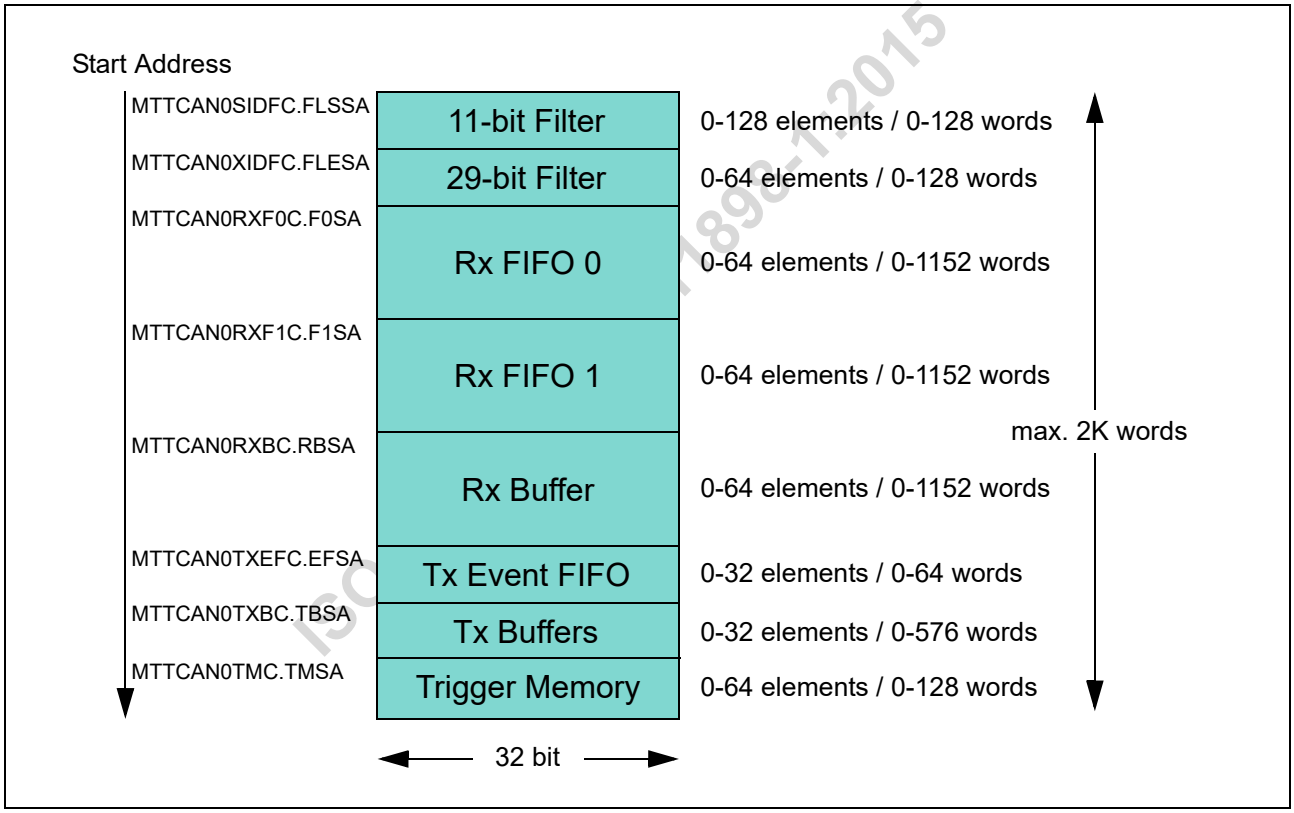



**Figure 19.16 Message RAM Configuration**

When the M_TTCAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

**NOTE**

The M_TTCAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

### (2) Rx Buffer and FIFO Element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in **Table 19.136** below. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register MTTCAN0RXESC.

**Table 19.136 Rx Buffer and FIFO Element**

| | 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|---|
| R0 | ESI / XTD / RTR / ID[28:0] | | | |
| R1 | ANMF / FIDX[6:0] | res / FDF / BRS / DLC[3:0] | RXTS[15:0] | |
| R2 | DB3[7:0] | DB2[7:0] | DB1[7:0] | DB0[7:0] |
| R3 | DB7[7:0] | DB6[7:0] | DB5[7:0] | DB4[7:0] |
| ... | ... | ... | ... | ... |
| Rn | DBm[7:0] | DBm-1[7:0] | DBm-2[7:0] | DBm-3[7:0] |

**R0 Bit 31 ESI: Error State Indicator**

0: Transmitting node is error active
1: Transmitting node is error passive

**R0 Bit 30 XTD: Extended Identifier**

Signals to the Host whether the received frame has a standard or extended identifier.

0: 11-bit standard identifier
1: 29-bit extended identifier

**R0 Bit 29 RTR: Remote Transmission Request**

Signals to the Host whether the received frame is a data frame or a remote frame.

0:Received frame is a data frame

1:Received frame is a remote frame

**NOTE**

There are no remote frames in CAN FD format. In case a CAN FD frame (FDF = '1'), the dominant RRS (Remote Request Substitution) bit replaces bit RTR (Remote Transmission Request).

ISO CANFD ISO 11898-1:2015

**R0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

**R1 Bit 31 ANMF: Accepted Non-matching Frame**

Acceptance of non-matching frames may be enabled via MTTCAN0GFC.ANFS and MTTCAN0GFC.ANFE.

0:Received frame matching filter index FIDX

1:Received frame did not match any Rx filter element

**R1 Bits 30:24 FIDX[6:0]:Filter Index**

0 to 127:Index of matching Rx acceptance filter element (invalid if ANMF = '1').
Range is 0 to MTTCAN0SIDFC.LSS - 1 resp. MTTCAN0XIDFC.LSE - 1.

**R1 Bit 21 FDF: FD Format**

0:Standard frame format

1:CAN FD frame format (new DLC-coding and CRC)

**R1 Bit 20 BRS: Bit Rate Switch**

0:Frame received without bit rate switching

1:Frame received with bit rate switching

**R1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: received frame has 0 to 8 data bytes

9 to 15: CAN: received frame has 8 data bytes

9 to 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes

**R1 Bits 15:0 RXTS[15:0]:Rx Timestamp**

Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler MTTCAN0TSCC.TCP.

ISO CANFD ISO 11898-1:2015

**R2 Bits 31:24 DB3[7:0]:Data Byte 3**

**R2 Bits 23:16 DB2[7:0]:Data Byte 2**

**R2 Bits 15:8 DB1[7:0]:Data Byte 1**

**R2 Bits 7:0 DB0[7:0]:Data Byte 0**

**R3 Bits 31:24 DB7[7:0]:Data Byte 7**

**R3 Bits 23:16 DB6[7:0]:Data Byte 6**

**R3 Bits 15:8 DB5[7:0]:Data Byte 5**

**R3 Bits 7:0 DB4[7:0]:Data Byte 4**

...

**Rn Bits 31:24 DBm[7:0]:Data Byte m**

**Rn Bits 23:16 DBm-1[7:0]:Data Byte m-1**

**Rn Bits 15:8 DBm-2[7:0]:Data Byte m-2**

**Rn Bits 7:0 DBm-3[7:0]:Data Byte m-3**

**NOTE**

Depending on the configuration of the element size (MCTTAN0RXESC), between two and sixteen 32-bit words (Rn = 3 to 17) are used for storage of a CAN message's data field.

ISO CAN FD ISO 11898-1:2015

### (3) Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration MTTCAN0TXBC.TFQS and MTTCAN0TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register MTTCAN0TXESC.

**Table 19.137 Tx Buffer Element**

| | 31 | 30 | 29 | 28 … 24 | 23 … 16 | 15 … 8 | 7 … 0 |
|---|---|---|---|---|---|---|---|
| T0 | ESI | XTD | RTR | ID[28:0] | | | |
| T1 | MM[7:0] | | | | EFC / res / FDF / BRS / DLC[3:0] | res | |
| T2 | DB3[7:0] | | | | DB2[7:0] | DB1[7:0] | DB0[7:0] |
| T3 | DB7[7:0] | | | | DB6[7:0] | DB5[7:0] | DB4[7:0] |
| ... | ... | | | | ... | ... | ... |
| Tn | DBm[7:0] | | | | DBm-1[7:0] | DBm-2[7:0] | DBm-3[7:0] |

**T0 Bit 31 ESI: Error State Indicator**

0: ESI bit in CAN FD format depends only on error passive flag

1: ESI bit in CAN FD format transmitted recessive

**NOTE**

The ESI bit of the transmit buffer is or'ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive

**T0 Bit 30 XTD: Extended Identifier**

0: 11-bit standard identifier

1: 29-bit extended identifier

**T0 Bit 29 RTR: Remote Transmission Request**

0: Transmit data frame

1: Transmit remote frame

ISO CANFD ISO 11898-1:2015

**NOTE**

When RTR = 1, the M_TTCAN transmits a remote frame according to ISO11898-1:2015, even if MTTCAN0CCCR.FDOE enables the transmission in CAN FD format.

**T0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

**T1 Bits 31:24 MM[7:0]: Message Marker**

Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.

**T1 Bit 23 EFC: Event FIFO Control**

0: Don't store Tx events

1: Store Tx events

**T1 Bit 21 FDF: FD Format**

0: Frame transmitted in Classic CAN format

1: Frame transmitted in CAN FD format

**T1 Bit 20 BRS: Bit Rat Switching**

0: CAN FD frames transmitted without bit rate switching

1: CAN FD frames transmitted with bit rate switching

**NOTE**

Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled MTTCAN0CCCR.FDOE = '1'. Bit BRS is only evaluated when in addition CCCR.BRSE = '1'.

**T1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: transmit frame has 0-8 data bytes

9 to 15: CAN: transmit frame has 8 data bytes

9 to 15: CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes

ISO CANFD ISO 11898-1:2015

**T2 Bits 31:24 DB3[7:0]:Data Byte 3**

**T2 Bits 23:16 DB2[7:0]: Data Byte 2**

**T2 Bits 15:8 DB1[7:0]: Data Byte 1**

**T2 Bits 7:0 DB0[7:0]: Data Byte 0**

**T3 Bits 31:24 DB7[7:0]: Data Byte 7**

**T3 Bits 23:16 DB6[7:0]: Data Byte 6**

**T3 Bits 15:8 DB5[7:0]: Data Byte 5**

**T3 Bits 7:0 DB4[7:0]: Data Byte 4**

**...**

**Tn Bits 31:24 DBm[7:0]: Data Byte m**

**Tn Bits 23:16 DBm-1[7:0]:Data Byte m-1**

**Tn Bits 15:8 DBm-2[7:0]:Data Byte m-2**

**Tn Bits 7:0 DBm-3[7:0]:Data Byte m-3**

**NOTE**

Depending on the configuration of the element size (MTTCAN0TXESC), between two and sixteen 32-bit words (Tn = 3 to 17) are used for storage of a CAN message's data field.

ISO CANFD ISO 11898-1:2015

### (4) Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register MTTCAN0TXEFS.

**Table 19.138 Tx Event FIFO Element**

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E0 | ESI | XTD | RTR | ID[28:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E1 | MM[7:0] | | | | | | | | ET[1:0] | | FDF | BRS | DLC[3:0] | | | | TXTS[15:0] | | | | | | | | | | | | | | | |

**E0 Bit 31 ESI: Error State Indicator**

0: Transmitting node is error active

1: Transmitting node is error passive

**E0 Bit 30 XTD: Extended Identifier**

0: 11-bit standard identifier

1: 29-bit extended identifier

**E0 Bit 29 RTR: Remote Transmission Request**

0: Data frame transmitted

1: Remote frame transmitted

**E0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

**E1 Bits 31:24 MM[7:0]: Message Marker**

Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.

**E1 Bit 23:22 ET[1:0]: Event Type**

00: Reserved

01: Tx event

10: Transmission in spite of cancellation (always set for transmissions in DAR mode)

11: Reserved

**E1 Bit 21 FDF: FD Format**

0: Standard frame format

1: CAN FD frame format (new DLC-coding and CRC)

**E1 Bit 20 BRS: Bit Rate Switch**

0: Frame transmitted without bit rate switching

1: Frame transmitted with bit rate switching

ISO CANFD ISO 11898-1:2015

**E1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: frame with 0-8 data bytes transmitted

9 to 15: CAN: frame with 8 data bytes transmitted

9 to 15: CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted

**E1 Bits 15:0 TXTS[15:0]:Tx Timestamp**

Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler MTTCAN0TSCC.TCP.

ISO CANFD ISO 11898-1:2015

### (5) Standard Message ID Filter Element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address MTTCAN0SIDFC.FLSSA plus the index of the filter element (0 to 127).

**Table 19.139 Standard Message ID Filter Element**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | SFT[1:0] | | SFEC[2:0] | | SFID1[10:0] | | | | | | | | | | | res | | | | | | | | SFID2[10:0] | | | | | | | |

**Bits 31:30 SFT[1:0]: Standard Filter Type**

00: Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1)

01: Dual ID filter for SFID1 or SFID2

10: Classic filter: SFID1 = filter, SFID2 = mask

11: Filter element disabled

**NOTE**

With SFT = “11” the filter element is disabled and the acceptance filtering continues (same behaviour as with SFEC = “000”).

**Bit 29:27 SFEC[2:0]:Standard Filter Element Configuration**

All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag MTTCAN0IR.HPM and, if enabled, an interrupt is generated. In this case register MTTCAN0HPMS is updated with the status of the priority match.

000: Disable filter element

001: Store in Rx FIFO 0 if filter matches

010: Store in Rx FIFO 1 if filter matches

011: Reject ID if filter matches

100: Set priority if filter matches

101: Set priority and store in FIFO 0 if filter matches

110: Set priority and store in FIFO 1 if filter matches

111: Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored

**Bits 26:16 SFID1[10:0]:Standard Filter ID 1**

First ID of standard ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.

ISO CANFD ISO 11898-1:2015

**Bits 10:0 SFID2[10:0]:Standard Filter ID 2**

This bit field has a different meaning depending on the configuration of SFEC:

1) SFEC = “001” to ”110” Second ID of standard ID filter element

2) SFEC = “111” Filter for Rx Buffers or for debug messages

**SFID2[10:9]**: These bits decide whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.
00: Store message into an Rx Buffer
01: Debug Message A
10: Debug Message B
11: Debug Message C

**SFID2[8:6]**: These bits are used to control the filter event pins m_ttcan_fe[2:0] at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_ttcan_hclk (CLK_HSB) period in case the filter matches.

**SFID2[5:0]:** These bits define the offset to the Rx Buffer Start Address MTTCAN0RXBC.RBSA for storage of a matching message.

ISO CANFD ISO 11898-1:2015

### (6) Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address MTTCAN0XIDFC.FLESA plus two times the index of the filter element (0 to 63).

**Table 19.140 Extended Message ID Filter Element**

| | 31 | 24 | 23 | 16 | 15 | 8 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|
| F0 | EFEC[2:0] | | EFID1[28:0] | | | | | |
| F1 | EFT[1:0] | res | EFID2[28:0] | | | | | |

#### F0 Bit 31:29 EFEC[2:0]:Extended Filter Element Configuration

All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = “100”, “101”, or “110” a match sets interrupt flag MCANnIR.HPM and, if enabled, an interrupt is generated. In this case register MCANnHPMS is updated with the status of the priority match.

000: Disable filter element

001: Store in Rx FIFO 0 if filter matches

010: Store in Rx FIFO 1 if filter matches

011: Reject ID if filter matches

100: Set priority if filter matches

101: Set priority and store in FIFO 0 if filter matches

110: Set priority and store in FIFO 1 if filter matches

111: Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored

ISO CANFD ISO 11898-1:2015

#### F0 Bits 28:0 EFID1[28:0]:Extended Filter ID 1

First ID of extended ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only MTTCAN0XIDAM masking mechanism (see **Section (e), Extended Message ID Filtering**) is used.

#### F1 Bits 31:30 EFT[1:0]: Extended Filter Type

00: Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1)

01: Dual ID filter for EFID1 or EFID2

10: Classic filter: EFID1 = filter, EFID2 = mask

11: Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), MTTCAN0XIDAM mask not applied

#### F1 Bits 28:0 EFID2[28:0]:Extended Filter ID 2

This bit field has a different meaning depending on the configuration of EFEC:

1) EFEC: “001” to ”110”Second ID of extended ID filter element

2) EFEC: “111”Filter for Rx Buffers or for debug messages

**EFID2[10:9]**: These bits decide whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.
00: Store message into an Rx Buffer
01: Debug Message A
10: Debug Message B
11: Debug Message C

**EFID2[8:6]**: These bits are used to control the filter event pins m_ttcan_fe[2:0] at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_ttcan_hclk (CLK_HSB) period in case the filter matches.

**EFID2[5:0]:** These bits define the offset to the Rx Buffer Start Address MCANnRXBC.RBSA for storage of a matching message.

ISO CANFD ISO 11898-1:2015

### (7) Trigger Memory Element

Up to 64 trigger memory elements can be configured. When accessing a Trigger Memory element, its address is the Trigger Memory Start Address MTTCAN0TTTMC.TMSA plus the index of the trigger memory element (0 to 63).

**Table 19.141 Trigger Memory Element**

| | 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|---|
| T0 | TM[15:0] | | res CC[6:0] | ASC[1:0] TMIN TMEX TYPE[3:0] |
| T1 | res | FTYPE MNR[6:0] | res | MSC[2:0] |

#### T0 Bit 31:16 TM[15:0]:Time Mark

Cycle time for which the trigger becomes active.

#### T0 Bit 14:8 CC[6:0]:Cycle Code

Cycle count for which the trigger is valid. Ignored for trigger types Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, End_of_List.

$000000x_B$ valid for all cycles

$000001c_B$ valid every 2nd cycle at cycle count mod2 = c

$00001cc_B$ valid every 4th cycle at cycle count mod4 = cc

$0001ccc_B$ valid every 8th cycle at cycle count mod8 = ccc

$001cccc_B$ valid every 16th cycle at cycle count mod16 = cccc

$01ccccc_B$ valid every 32nd cycle at cycle count mod32 = ccccc

$1cccccc_B$ valid every 64th cycle at cycle count mod64 = cccccc

#### T0 Bit 7:6 ASC[1:0]:Asynchronous Serial Communication

00: No ASC operation

01: Reserved, do not use

10: Node is ASC receiver

11: Node is ASC transmitter

#### T0 Bit 5 TMIN:Time Mark Event Internal

0: No action

1: MTTCAN0TTIR.TTMI is set when trigger memory element becomes active

#### T0 Bit 4 TMEX:Time Mark Event External

0: No action

1: Pulse at output m_ttcan_tmp (MTTCAN0TMP) with the length of one m_ttcan_cclk (CLKP_H2) period is generated when the time mark of the trigger memory element becomes active and MTTCAN0TTOCN.TTMIE = '1'

ISO CANFD ISO 11898-1:2015

**T0 Bit 3:0 TYPE[3:0]:Trigger Type**

| | |
|---|---|
| 0000 | Tx_Ref_Trigger - valid when not in Gap |
| 0001 | Tx_Ref_Trigger_Gap - valid when in Gap |
| 0010 | Tx_Trigger_Single - starts a single transmission in an exclusive time window |
| 0011 | Tx_Trigger_Continuous - starts continuous transmission in an exclusive time window |
| 0100 | Tx_Trigger_Arbitration - starts a transmission in an arbitrating time window |
| 0101 | Tx_Trigger_Merged - starts a merged arbitration window |
| 0110 | Watch_Trigger - valid when not in Gap |
| 0111 | Watch_Trigger_Gap - valid when in Gap |
| 1000 | Rx_Trigger - check for reception |
| 1001 | Time_Base_Trigger - only control TMIN, TMEX, and ASC |
| 1010 to 1111 | End_of_List - illegal type, causes config error |

**NOTE**

For ASC operation (ASC = “10”, “11”) only trigger types Rx_Trigger and Time_Base_Trigger should be used.

**T1 Bit 23 FTYPE:Filter Type**

*0*= 11-bit standard message ID
*1*= 29-bit extended message ID

**T1 Bit 22:16 MNR[6:0]:Message Number**

Transmission: Trigger is valid for configured Tx Buffer number. Valid values are 0 to 31.
Reception: Trigger is valid for standard / extended message ID filter element number. Valid values are 0 to 63 resp. 0 to 127.

**T1 Bits 2:0 MSC[2:0]:Message Status Count**

Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1:2015).

0-7: Actual status

**NOTES**

1. The trigger memory elements have to be written when the M_TTCAN is in INIT state. Write access to the trigger memory elements outside INIT state is not allowed.
2. There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx_Ref_Trigger. In this case they become active at the time mark modified by the actual Reference Trigger Offset (MTTCAN0TTOST.RTO).

ISO CANFD ISO 11898-1:2015

## 19.6.3 Functional Description

### 19.6.3.1 Operating Modes

#### (1) Software Initialization

Software initialization is started by setting bit MTTCAN0CCCR.INIT, either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While MTTCAN0CCCR.INIT is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output m_ttcan_tx is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting MTTCAN0CCCR.INIT does not change any configuration register. Resetting MTTCAN0CCCR.INIT finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (≡ Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_TTCAN configuration registers is only enabled when both bits MTTCAN0CCCR.INIT and MTTCAN0CCCR.CCE are set (protected write).

MTTCAN0CCCR.CCE can only be set/reset while MTTCAN0CCCR.INIT = '1'. MTTCAN0CCCR.CCE is automatically reset when MTTCAN0CCCR.INIT is reset.

The following registers are reset when MTTCAN0CCCR.CCE is set

- MTTCAN0HPMS - High Priority Message Status
- MTTCAN0RXF0S - Rx FIFO 0 Status
- MTTCAN0RXF1S - Rx FIFO 1 Status
- MTTCAN0TXFQS - Tx FIFO/Queue Status
- MTTCAN0TXBRP - Tx Buffer Request Pending
- MTTCAN0TXBTO - Tx Buffer Transmission Occurred
- MTTCAN0TXBCF - Tx Buffer Cancellation Finished
- MTTCAN0TXEFS - Tx Event FIFO Status
- MTTCAN0TTOST - TT Operation Status
- MTTCAN0TTLGT - TT Local & Global Time, only Global Time MTTCAN0TTLGT.GT is reset
- MTTCAN0TTCTC - TT Cycle Time & Count
- MTTCAN0TTCSM - TT Cycle Sync Mark

The Timeout Counter value MTTCAN0TOCV.TOC is preset to the value configured by MTTCAN0TOCC.TOP when MTTCAN0CCCR.CCE is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while MTTCAN0CCCR.CCE = '1'.

The following registers are only writable while MTTCAN0CCCR.CCE = '0'

- MTTCAN0TXBAR - Tx Buffer Add Request
- MTTCAN0TXBCR - Tx Buffer Cancellation Request

MTTCAN0CCCR.TEST and MTTCAN0CCCR.MON can only be set by the Host while MTTCAN0CCCR.INIT = '1' and MTTCAN0CCCR.CCE = '1'. Both bits may be reset at any time. MTTCAN0CCCR.DAR can only be set/reset while MTTCAN0CCCR.INIT = '1' and MTTCAN0CCCR.CCE = '1'.

ISO CANFD ISO 11898-1:2015

### (2) Normal Operation

The M_TTCAN's default operating mode after hardware reset is event-driven CAN communication without time triggers (MTTCAN0TTOCF.OM = "00"). It is required that both MTTCAN0CCCR.INIT and MTTCAN0CCCR.CCE are set before the TT Operation Mode can be changed.

Once the M_TTCAN is initialized and MTTCAN0CCCR.INIT is reset to zero, the M_TTCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

### (3) CAN FD Operation

There are two variants in the CAN FD frame transmission, first the CAN FD frame without bit rate switching . The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = recessive signifies a CAN FD frame, FDF = dominant signifies a Classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. The coding of res = recessive is reserved for future expansion of the protocol. In case the M_TTCAN receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting bit PSR.PXE. When Protocol Exception Handling is enabled (MTTCAN0CCCR.PXHD = '0'), this causes the operation state to change from Receiver (MTTCAN0PSR.ACT = "10") to Integrating (MTTCAN0PSR.ACT = "00") at the next sample point. In case Protocol Exception Handling is disabled (MTTCAN0CCCR.PXHD = '1'), the M_TTCAN will treat a recessive res bit as an form error and will respond with an error frame.

ISO CAN FD ISO 11898-1:2015

CAN FD operation is enabled by programming MTTCAN0CCCR.FDOE. In case MTTCAN0CCCR.FDOE = '1', transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via bit FDF in the respective Tx Buffer element. With MTTCAN0CCCR.FDOE = '0', received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx Buffer element is set. MTTCAN0CCCR.FDOE and MTTCAN0CCCR.BRSE can only be changed while MTTCAN0CCCR.INIT and CCCR.CCE are both set.

With MTTCAN0CCCR.FDOE = '0', the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With MTTCAN0CCCR.FDOE = '1' and MTTCAN0CCCR.BRSE = '0', only bit FDF of a Tx Buffer element is evaluated. With MTTCAN0CCCR.FDOE = '1' and MTTCAN0CCCR.BRSE = '1', transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting Classic CAN messages until it is verified that

they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.

- Wake-up messages in CAN Partial Networking have to be transmitted in Classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to **Table 19.142** below.

**Table 19.142 Coding of DLC in CAN FD**

| DLC | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| Number of Data Bytes | 12 | 16 | 20 | 24 | 32 | 48 | 64 |

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the Nominal Bit Timing & Prescaler Register MTTCAN0NBTP. In the following CAN FD data phase, the data phase bit timing is used as defined by the Data Bit Timing & Prescaler Register MTTCAN0DBTP. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (m_ttcan_cclk: CLKP_H2). Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD and CAN FD with bit rate switching, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

ISO CANFD ISO 11898-1:2015

### (4) Transmitter Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin m_ttcan_tx the protocol controller receives the transmitted data from its local CAN transceiver via pin m_ttcan_rx. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transceiver loop delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

(a) Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the ISO11898-1:2015. It is enabled by setting bit MTTCAN0DBTP.TDC.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the M_TTCAN's transmit output m_ttcan_tx through the transceiver to the receive input m_ttcan_rx plus the transmitter delay compensation offset as configured by MTTCAN0TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq.

MTTCAN0PSR.TDCV shows the actual transmitter delay compensation value. MTTCAN0PSR.TDCV is cleared when MTTCAN0CCCR.INIT is set and is updated at each transmission of an FD frame while MTTCAN0DBTP.TDC is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the M_TTCAN:

- The sum of the measured delay from m_ttcan_tx to m_ttcan_rx and the configured transmitter delay compensation offset MTTCAN0TDCR.TDCO has to be less than 6 bit times in the data phase.
- The sum of the measured delay from m_ttcan_tx to m_ttcan_rx and the configured transmitter delay compensation offset MTTCAN0TDCR.TDCO has to be less or equal 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs

ISO CANFD ISO 11898-1:2015

(b) Transmitter Delay Compensation Measurement

If transmitter delay compensation is enabled by programming MTTCAN0DBTP.TDC = '1', the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input m_ttcan_rx of the transmitter. The resolution of this measurement is one mtq.

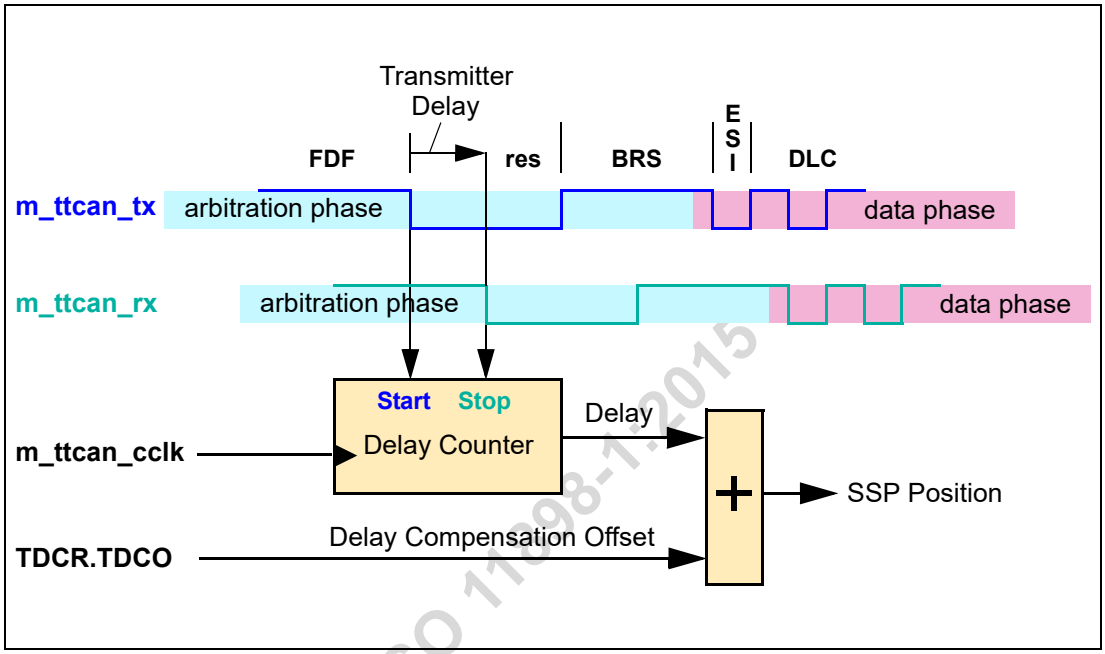



Figure 19.17 Transmitter delay measurement

ISO CANFD ISO 11898-1:2015

To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a to early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming MTTCAN0TDCR.TDCF. This defines a minimum value for the SSP position. Dominant edges on m_ttcan_rx, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least MTTCAN0TDCR.TDCF AND m_ttcan_rx is low.

### (5) Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters (ECR.REC, ECR.TEC) are frozen while Error Logging (ECR.CEL) is active. The Host can set the M_TTCAN into Restricted Operation mode by setting bit MTTCAN0CCCR.ASM. The bit can only be set by the Host when both MTTCAN0CCCR.CCE and MTTCAN0CCCR.INIT are set to '1'. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset MTTCAN0CCCR.ASM.

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

When the M_TTCAN is configured for Asynchronous Serial Communication, the Host has to set MTTCAN0CCCR.ASM during initialization to start with an ASC window. The bit is reset at the end of the first ASC window after the M_TTCAN has finished initializing. During time-triggered operation MTTCAN0CCCR.ASM is set by the M_TTCAN at the beginning of an ASC window (trigger memory element with T0.ASC = "10", "11"). It is reset by each trigger memory element with T0.ASC = "00".

If the M_TTCAN is connected to a Clock Calibration on CAN unit, MTTCAN0CCCR.ASM is controlled by input m_ttcan_cok. In case m_ttcan_cok switches to '0', bit MTTCAN0CCCR.ASM is set. When m_ttcan_cok switches back to '1', bit MTTCAN0CCCR.ASM returns to the previously written value. When there is no Clock Calibration on CAN unit connected input m_ttcan_cok is hardwired to '1'.

**NOTE**

The Restricted Operation Mode must not be combined with the Loop Back Mode (internal or external).

ISO CANFD ISO 11898-1:2015

### (6) Bus Monitoring Mode

The M_TTCAN is set in Bus Monitoring Mode by programming MTTCAN0CCCR.MON to one or when error level S3 (MTTCAN0TTOST.EL = “11”) is entered. In Bus Monitoring Mode (see ISO11898-1:2015, 10.12 Bus monitoring), the M_TTCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the M_TTCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_TTCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode register MTTCAN0TXBRP is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. **Figure 19.18** shows the connection of signals m_ttcan_tx and m_ttcan_rx to the M_TTCAN in Bus Monitoring Mode.

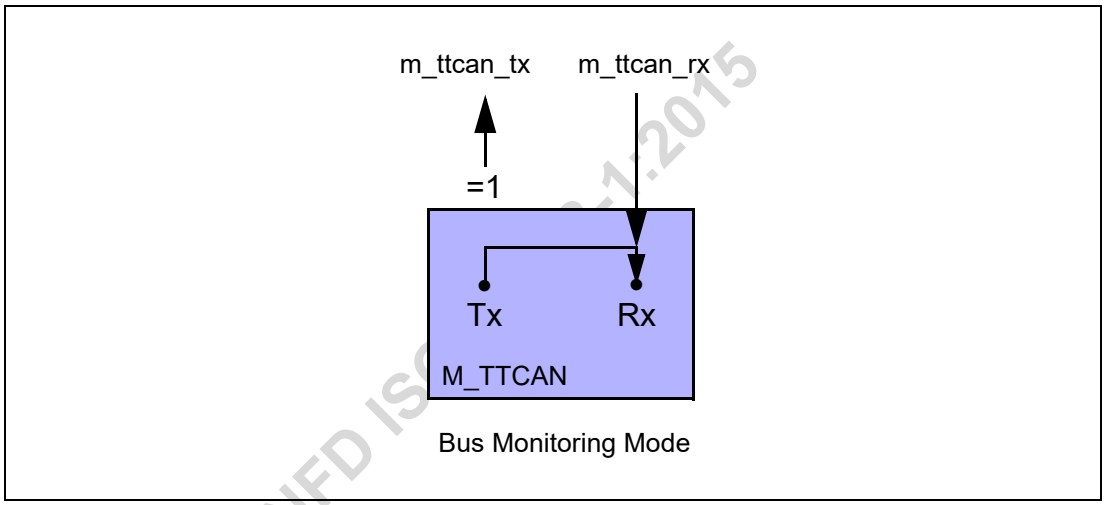



**Figure 19.18 Pin Control in Bus Monitoring Mode**

### (7) Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898-1:2015, 6.3.3 Recovery Management), the M_TTCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via MTTCAN0CCCR.DAR.

#### (a) Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer’s Tx Request Pending bit MTTCAN0TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  Corresponding Tx Buffer Transmission Occurred bit MTTCAN0TXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit MTTCAN0TXBCF.CFx not set
- Successful transmission in spite of cancellation:
  Corresponding Tx Buffer Transmission Occurred bit MTTCAN0TXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit MTTCAN0TXBCF.CFx set
- Arbitration lost or frame transmission disturbed:

ISO CANFD ISO11898-1:2015

Corresponding Tx Buffer Transmission Occurred bit MTTCAN0TXBTO.TOx not set
Corresponding Tx Buffer Cancellation Finished bit MTTCAN0TXBCF.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = “10” (transmission in spite of cancellation).

### (8) Power Down (Sleep Mode)

The M_TTCAN can be set into power down mode by using CC Control Register MTTCAN0CCCR.CSR. When all pending transmission requests have completed, the M_TTCAN waits until bus idle state is detected. Then the M_TTCAN sets then MTTCAN0CCCR.INIT to one to prevent any further CAN transfers. Now the M_TTCAN acknowledges that it is ready for power down by MTTCAN0CCCR.CSA to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to MTTCAN0CCCR.INIT will have no effect. Now the module clock inputs m_ttcan_hclk (CLK_HSB) and m_ttcan_cclk (CLKP_H2) may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting CC Control Register flag MTTCAN0CCCR.CSR. The M_TTCAN will acknowledge this by resetting MTTCAN0CCCR.CSA. Afterwards, the application can restart CAN communication by resetting bit MTTCAN0CCCR.INIT.

### (9) Test Modes

To enable write access to register MTTCAN0TEST (see **(4) MCANnTEST — Test Register**), bit MTTCAN0CCCR.TEST has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin m_ttcan_tx by programming MTTCAN0TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_TTCAN’s bit timing and it can drive constant dominant or recessive values. The actual value at pin m_ttcan_rx can be read from MTTCAN0TEST.RX. Both functions can be used to check the CAN bus’ physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to MTTCAN0TEST.TX until the new configuration is visible at output pin m_ttcan_tx. This applies also when reading input pin m_ttcan_rx via MTTCAN0TEST.RX.

ISO CAN FD ISO 11898-1:2015

**NOTE**

Test modes should be used for production tests or self test only. The software control for pin m_ttcan_tx interferes with all CAN protocol functions. It is not recommended to use test modes for application.

#### (a) External Loop Back Mode

The M_TTCAN can be set in External Loop Back Mode by programming MTTCAN0TEST.LBCK to one. In Loop Back Mode, the M_TTCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx Buffer or an Rx FIFO. **Figure 19.19** shows the connection of signals m_ttcan_tx and m_ttcan_rx to the M_TTCAN in External Loop Back Mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_TTCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the M_TTCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the m_ttcan_rx input pin is disregarded by the M_TTCAN. The transmitted messages can be monitored at the m_ttcan_tx pin.

(b) Internal Loop Back Mode

Internal Loop Back Mode is entered by programming bits MTTCAN0TEST.LBCK and MTTCAN0CCCR.MON to one. This mode can be used for a “Hot Selftest”, meaning the M_TTCAN can be tested without affecting a running CAN system connected to the pins m_ttcan_tx and m_ttcan_rx. In this mode pin m_ttcan_rx is disconnected from the M_TTCAN and pin m_ttcan_tx is held recessive. **Figure 19.19** shows the connection of m_ttcan_tx and m_ttcan_rx to the M_TTCAN in case of Internal Loop Back Mode.

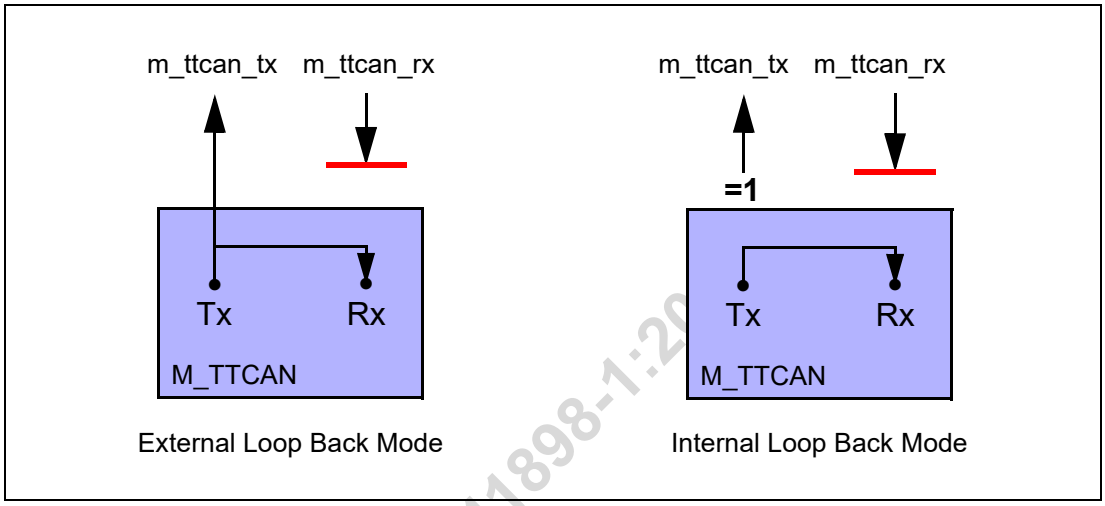



**Figure 19.19 Pin Control in Loop Back Modes**

ISO CAN FD ISO 11898-1:20

### (10) Application Watchdog

The application watchdog is served by reading register MTTCAN0TTOST. When the application watchdog is not served in time, bit MTTCAN0TTOST.AWE is set, all TTCAN communication is stopped, and the M_TTCAN is set into Bus Monitoring Mode.

The TT Application Watchdog can be disabled by programming the Application Watchdog Limit MTTCAN0TTOCF.AWL to $00_H$. The TT Application Watchdog should not be disabled in a TTCAN application program.

### 19.6.3.2 Timestamp Generation

For timestamp generation the M_TTCAN supplies a 16-bit wrap-around counter. A prescaler MTTCAN0TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via MTTCAN0TSCV.TCV. A write access to register MTTCAN0TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag MTTCAN0IR.TSW is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer / Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit MTTCAN0TSCC.TSS an external 16-bit timestamp can be used.

### 19.6.3.3 Timeout Counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_TTCAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by MTTCAN0TSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via register MTTCAN0TOCC. The actual counter value can be read from MTTCAN0TOCV.TOC.

The Timeout Counter can only be started while MTTCAN0CCCR.INIT = '0'. It is stopped when MTTCAN0CCCR.INIT = '1', e.g. when the M_TTCAN enters Bus_Off state.

The operation mode is selected by MTTCAN0TOCC.TOS. When operating in Continuous Mode, the counter starts when MTTCAN0CCCR.INIT is reset. A write to MTTCAN0TOCV presets the counter to the value configured by MTTCAN0TOCC.TOP and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by MTTCAN0TOCC.TOP. Down-counting is started when the first FIFO element is stored. Writing to MTTCAN0TOCV has no effect.

When the counter reaches zero, interrupt flag MTTCAN0IR.TOO is set. In Continuous Mode, the counter is immediately restarted at MTTCAN0TOCC.TOP.

**NOTE**

The clock signal for the Timeout Counter is derived from the CAN Core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

ISO CAN FD ISO 11898-1:2015

### 19.6.3.4 Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

#### (1) Acceptance Filtering

The M_TTCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration MTTCAN0GFC
- Standard ID Filter Configuration MTTCAN0SIDFC
- Extended ID Filter Configuration MTTCAN0XIDFC
- Extended ID AND Mask MTTCAN0XIDAM

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag MTTCAN0IR.HPM
- Set High Priority Message interrupt flag MTTCAN0IR.HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

ISO CAN FD ISO 11898-1:2015

### Rx Buffer

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see MTTCAN0PSR.LEC respectively MTTCAN0PSR.DLEC.

### Rx FIFO

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see MTTCAN0PSR.LEC respectively MTTCAN0PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in **(b) Rx FIFO Overwrite Mode** have to be considered.

**NOTE**

When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

(a) Range Filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

EFT: “00”: The Message ID of received frames is ANDed with the Extended ID AND Mask (MTTCAN0XIDAM) before the range filter is applied

EFT: “11”: The Extended ID AND Mask (MTTCAN0XIDAM) is not used for range filtering

(b) Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

(c) Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

ISO CAN FD ISO 11898-1:2015

(d) Standard Message ID Filtering

**Figure 19.20** below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in **(5) Standard Message ID Filter Element**.

Controlled by the Global Filter Configuration MTTCAN0GFC and the Standard ID Filter Configuration MTTCAN0SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

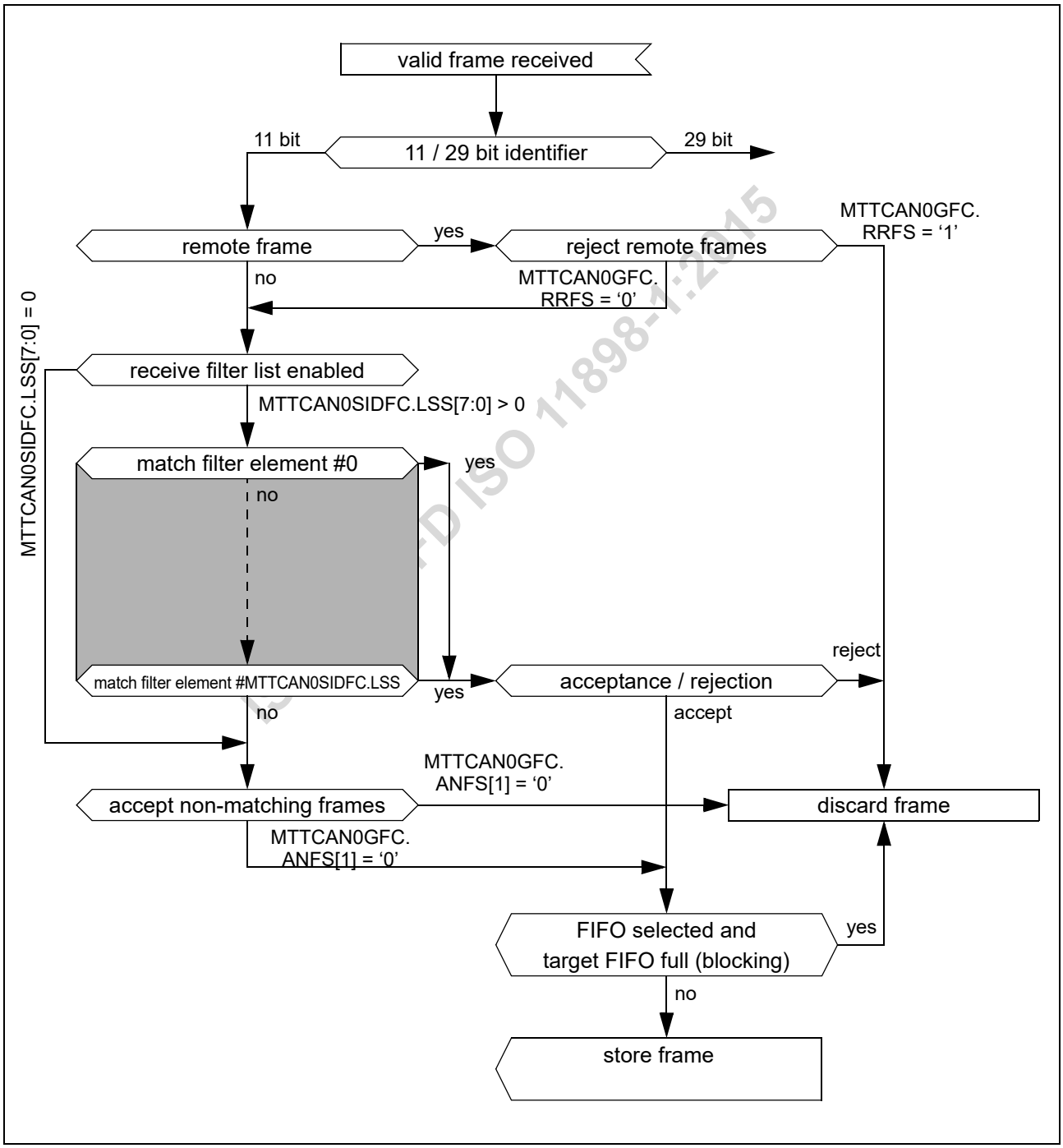



**Figure 19.20 Standard Message ID Filter Path**

(e) Extended Message ID Filtering

**Figure 19.21** below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in **(6) Extended Message ID Filter Element**.

Controlled by the Global Filter Configuration MTTCAN0GFC and the Extended ID Filter Configuration MTTCAN0XIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask MTTCAN0XIDAM is ANDed with the received identifier before the filter list is executed.

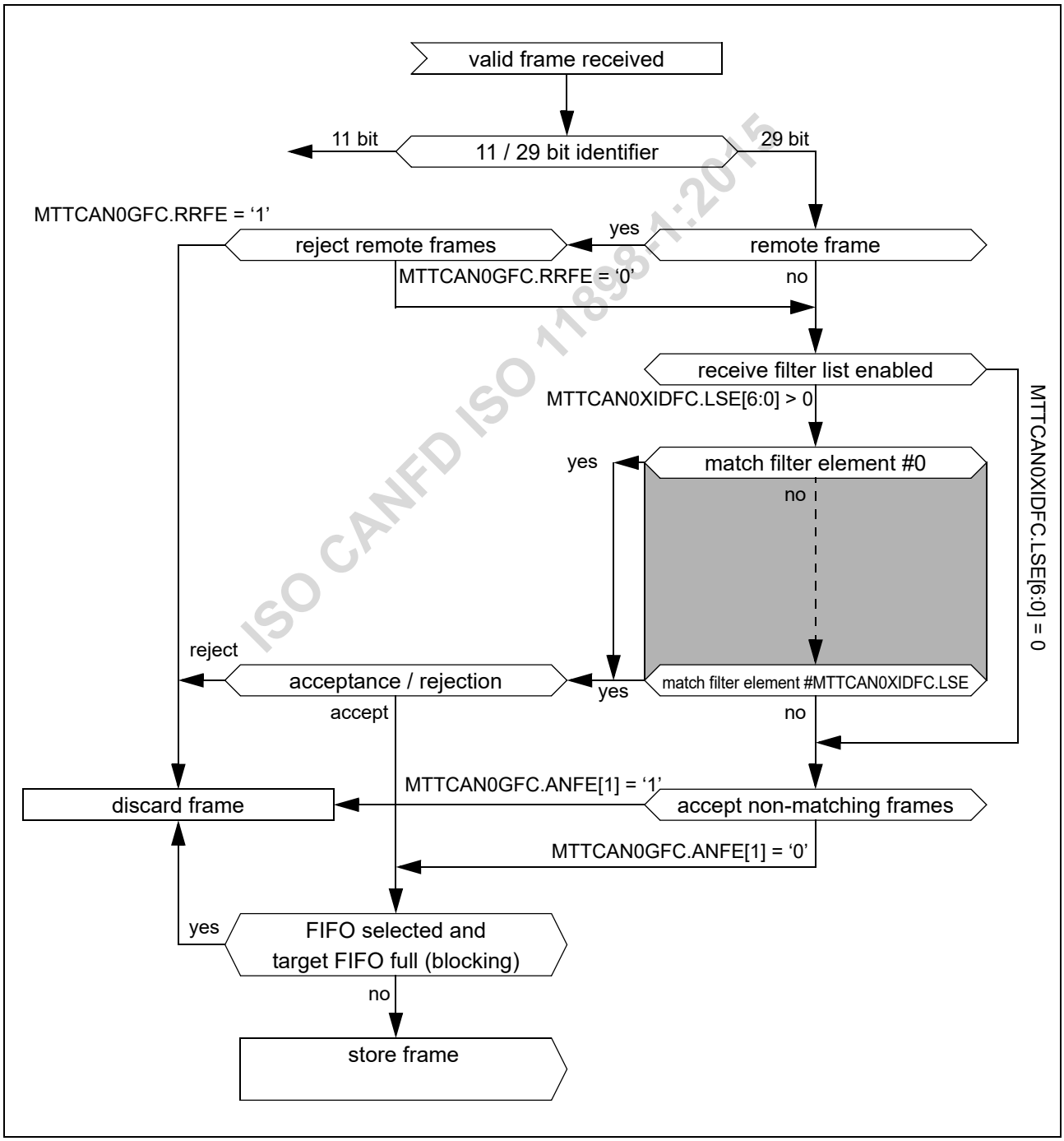



**Figure 19.21 Extended Message ID Filter Path**

### (2) Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers MTTCAN0RXF0C and MTTCAN0RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see **(1) Acceptance Filtering**. The Rx FIFO element is described in **(2) Rx Buffer and FIFO Element**.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag MTTCAN0IR.RFnW is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS.FnF. In addition interrupt flag MTTCAN0IR.RFnF is set.

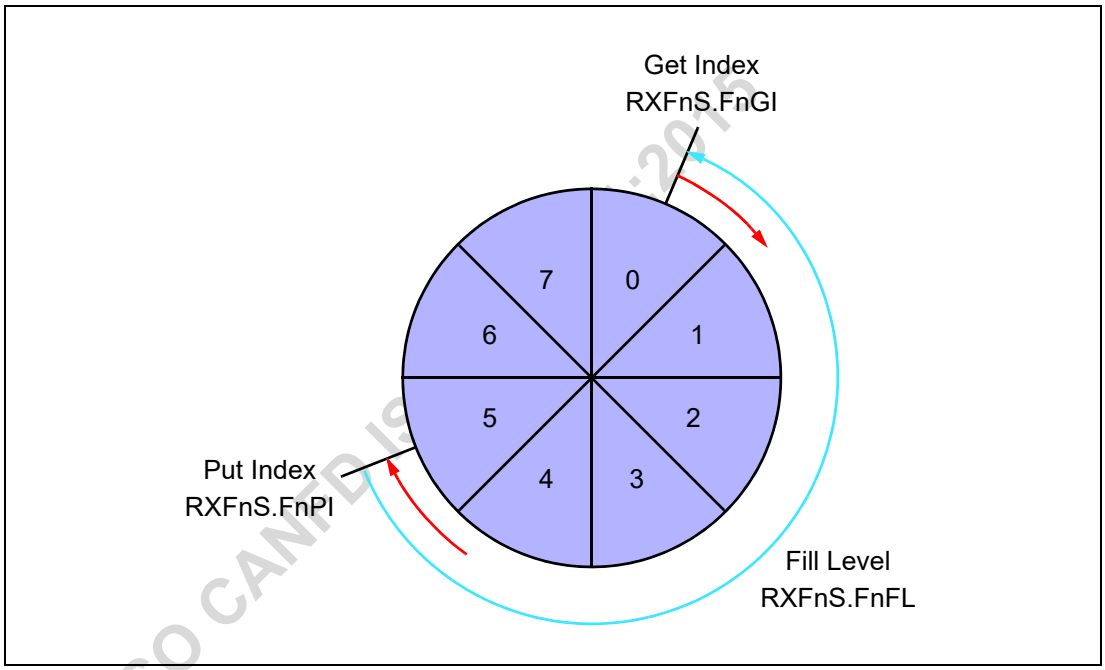



Figure 19.22 Rx FIFO Status

When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI • FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

Table 19.143 Rx Buffer / FIFO Element Size

| MTTCAN0RXESC.RBDS[2:0]<br>MTTCAN0RXESC.FnDS[2:0] | Data Field<br>[bytes] | FIFO Element Size<br>[RAM words] |
|---|---|---|
| 000 | 8 | 4 |
| 001 | 12 | 5 |
| 010 | 16 | 6 |
| 011 | 20 | 7 |
| 100 | 24 | 8 |
| 101 | 32 | 10 |
| 110 | 48 | 14 |
| 111 | 64 | 18 |

(a) Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = '0'. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS.FnF = '1'. In addition interrupt flag MTTCAN0IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = '1'. In addition interrupt flag MTTCAN0IR.RFnL is set.

(b) Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = '1'.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signalled by RXFnS.FnF = '1', the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. **Figure 19.23** shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

ISO CANFD ISO 11898-1:2015

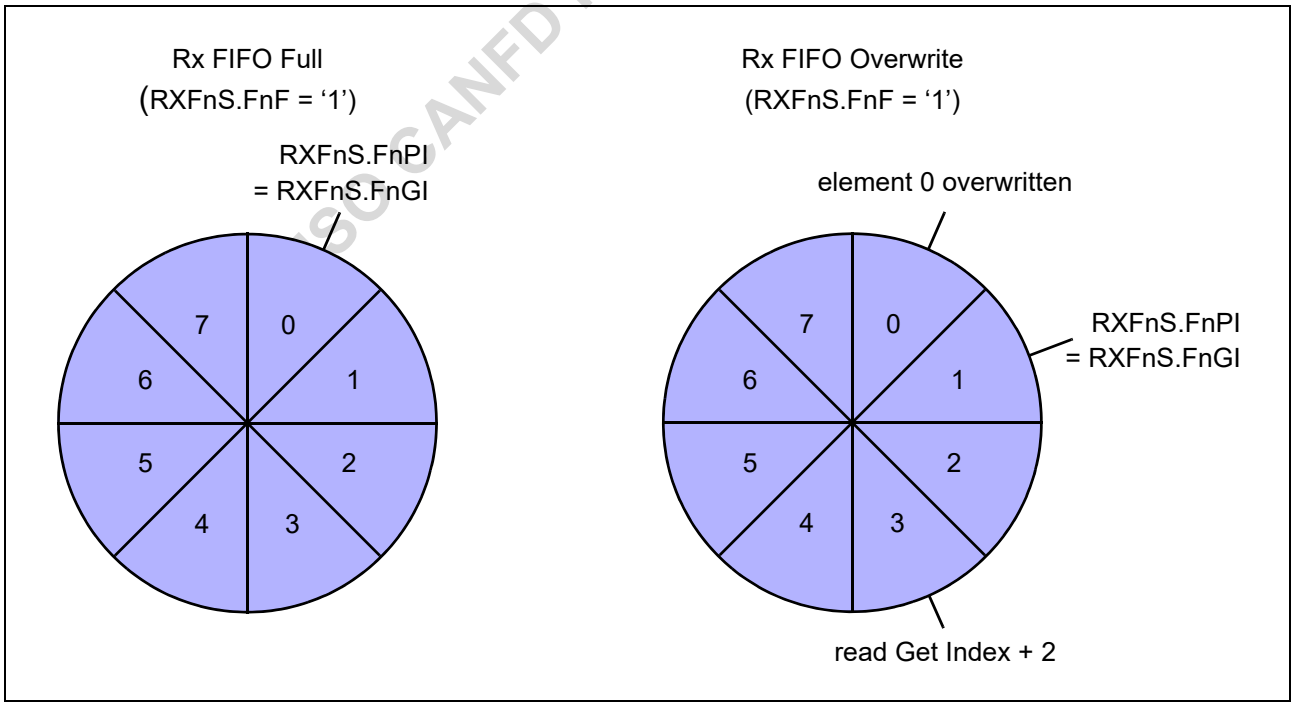



Figure 19.23 Rx FIFO Overflow Handling

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = '0').

### (3) Dedicated Rx Buffers

The M_TTCAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via MTTCAN0RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = “111” and SFID2 / EFID2[10:9] = “00” has to be configured (see **(5) Standard Message ID Filter Element** and **(6) Extended Message ID Filter Element**).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag MTTCAN0IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

**Table 19.144 Example Filter Configuration for Rx Buffers**

| Filter Element | SFID1[10:0] EFID1[28:0] | SFID2[10:9] EFID2[10:9] | SFID2[5:0] EFID2[5:0] |
|---|---|---|---|
| 0 | ID message 1 | 00 | 00 0000 |
| 1 | ID message 2 | 00 | 00 0001 |
| 2 | ID message 3 | 00 | 00 0010 |

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register MTTCAN0NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a ‘1’ to the respective bit position.

While an Rx Buffer’s New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

(a) Rx Buffer Handling

- Reset interrupt flag MTTCAN0IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

### (4) Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (e.g. #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (see M_TTCAN User’s Manual section 2.4.2).

Advantage: Fixed start address for the DMA transfers (relative to MTTCAN0RXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = “111” have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output m_ttcan_dma_req is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M_TTCAN while m_ttcan_dma_req is activated. The behaviour is similar to that of an Rx Buffers with its New Data flag set.

ISO CANFD ISO 11898-1:2015

After the DMA has completed the DMA unit sets m_ttcan_dma_ack. This resets m_ttcan_dma_req. Now the M_TTCAN is prepared to receive the next set of debug messages.

### (a) Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to “111”. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning (see Section 2.4.5 and Section 2.4.6). While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor MTTCAN0IR.DRX are set. The reception of debug messages can be monitored via MTTCAN0RXF1S.DMS

**Table 19.145 Example Filter Configuration for Debug Message**

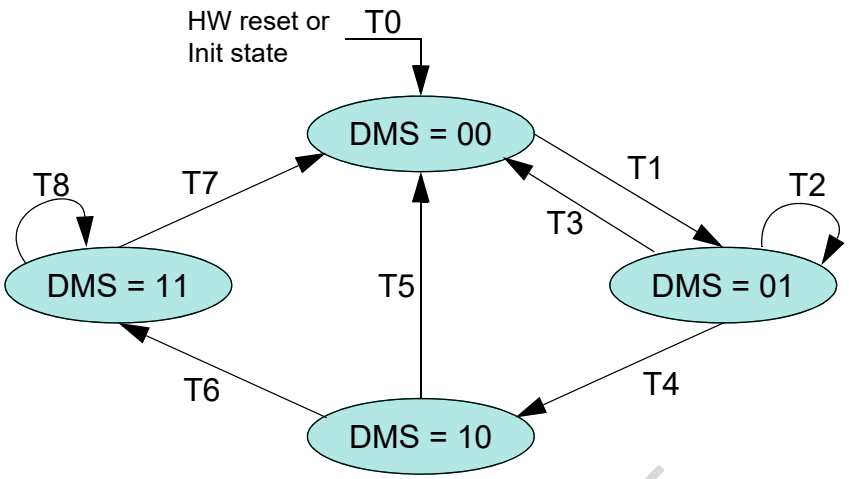
| Filter Element | SFID1[10:0] EFID1[28:0] | SFID2[10:9] EFID2[10:9] | SFID2[5:0] EFID2[5:0] |
|---|---|---|---|
| 0 | ID debug message A | 01 | 11 1101 |
| 1 | ID debug message B | 10 | 11 1110 |
| 2 | ID debug message C | 11 | 11 1111 |

### (b) Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signalled via RXF1S.DMS

ISO CANFD ISO 11898-1:201




T0: reset m_ttcan_dma_req output, enable reception of debug messages A, B, and C
T1: reception of debug message A
T2: reception of debug message A
T3: reception of debug message C
T4: reception of debug message B
T5: reception of debug messages A, B
T6: reception of debug message C
T7: DMA transfer completed
T8: reception of debug message A,B,C (message rejected)

ISO CANFD ISO 11898-1:2015

**Figure 19.24 Debug Message Handling State Machine**

### 19.6.3.5 Tx Handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices,and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The CAN mode for transmission (Classic CAN or CAN FD) can be configured separately for each Tx Buffer element. The Tx Buffer element is described in **(3) Tx Buffer Element**. **Table 19.146** below describes the possible configurations for frame transmission

**Table 19.146 Possible Configurations for Frame Transmission**

| CCCR | | Tx Buffer Element | | |
|---|---|---|---|---|
| BRSE | FDOE | FDF | BRS | Frame Transmission |
| ignored | 0 | ignored | ignored | Classic CAN |
| 0 | 1 | 0 | ignored | Classic CAN |
| 0 | 1 | 1 | ignored | FD without bit rate switching |
| 1 | 1 | 0 | ignored | Classic CAN |
| 1 | 1 | 1 | 0 | FD without bit rate switching |
| 1 | 1 | 1 | 1 | FD with bit rate switching |

**NOTE**

AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register MTTCAN0TXBRP is updated, or when a transmission has been started.

#### (1) Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit MTTCAN0CCCR.TXP. If the bit is set, the M_TTCAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (MTTCAN0CCCR.TXP = '0').

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

ISO CANFD ISO 11898-1:2015

### (2) Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via MTTCAN0TXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (see **Table 19.147**). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0 to 31) • Element Size to the Tx Buffer Start Address MTTCAN0TXBC.TBSA.

**Table 19.147 Tx Buffer / FIFO / Queue Element Size**

| MTTCAN0TXESC.TBDS[2:0] | Data Field [bytes] | Element Size [RAM words] |
|---|---|---|
| 000 | 8 | 4 |
| 001 | 12 | 5 |
| 010 | 16 | 6 |
| 011 | 20 | 7 |
| 100 | 24 | 8 |
| 101 | 32 | 10 |
| 110 | 48 | 14 |
| 111 | 64 | 18 |

ISO CANFD ISO 11898-1:20

### (3) Tx FIFO

Tx FIFO operation is configured by programming MTTCAN0TXBC.TFQM to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index MTTCAN0TXFQS.TFGI. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_TTCAN calculates the Tx FIFO Free Level MTTCAN0TXFQS.TFFL as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index MTTCAN0TXFQS.TFQPI. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (MTTCAN0TXFQS.TFQF = '1') is signalled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the MTTCAN0TXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via MTTCAN0TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see **Table 19.147**). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index MTTCAN0TXFQS.TFQPI (0 to 31) • Element Size to the Tx Buffer Start Address MTTCAN0TXBC.TBSA.

ISO CANFD ISO 11898-1:2015

### (4) Tx Queue

Tx Queue operation is configured by programming MTTCAN0TXBC.TFQM to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index MTTCAN0TXFQS.TFQPI. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (MTTCAN0TXFQS.TFQF = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register MTTCAN0TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see **Table 19.147**). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index MTTCAN0TXFQS.TFQPI (0 to 31) • Element Size to the Tx Buffer Start Address MTTCAN0TXBC.TBSA.

### (5) Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by MTTCAN0TXBC.NDTB. The number of Tx Buffers assigned to the Tx FIFO is configured by MTTCAN0TXBC.TFQS. In case MTTCAN0TXBC.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

ISO CANFD ISO 11898-1:2015

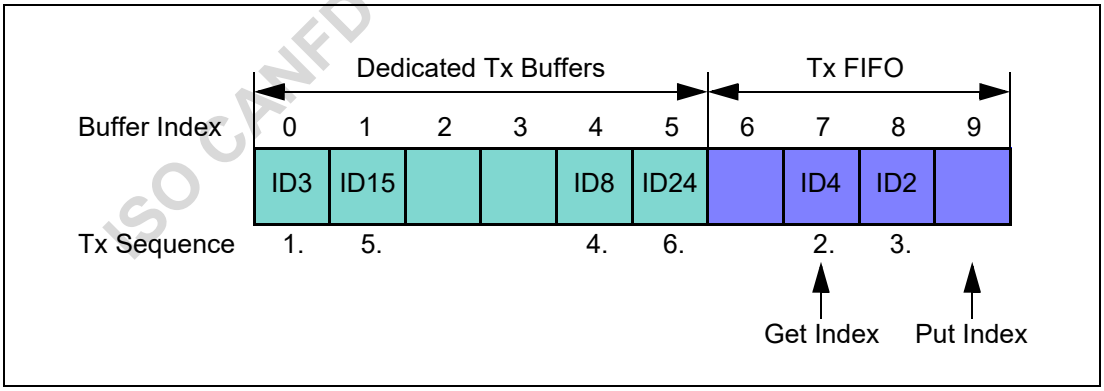



**Figure 19.25 Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO**

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS.TFGI)
- Buffer with lowest Message ID gets highest priority and is transmitted next

### (6) Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by MTTCAN0TXBC.NDTB. The number of Tx Queue Buffers is configured by MTTCAN0TXBC.TFQS. In case MTTCAN0TXBC.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

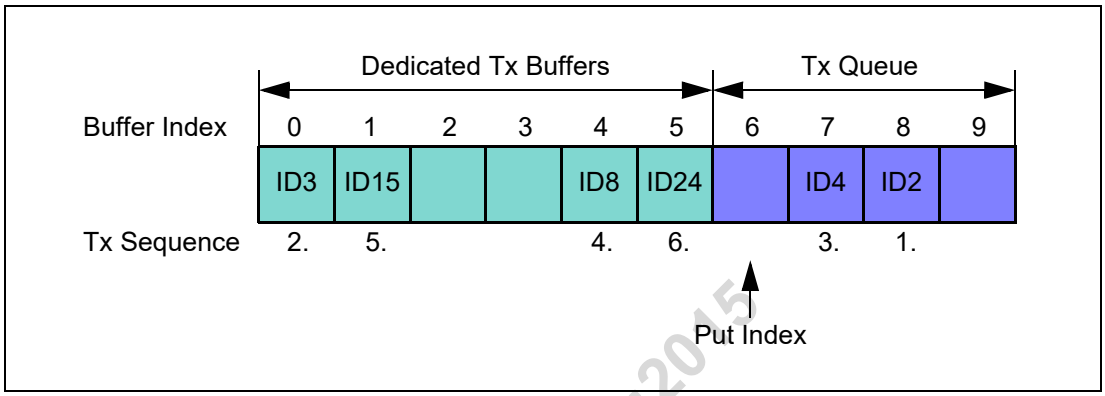



**Figure 19.26 Example of mixed Configuration Dedicated Tx Buffers / Tx Queue**

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

### (7) Transmit Cancellation

The M_TTCAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a Dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register MTTCAN0TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signalled by setting the corresponding bit of register MTTCAN0TXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding MTTCAN0TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding MTTCAN0TXBTO and MTTCAN0TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding MTTCAN0TXBCF bit is set.

**NOTE**

In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

ISO CAN FD ISO 11898-1:2015

### (8) Tx Event Handling

To support Tx event handling the M_TTCAN has implemented a Tx Event FIFO. After the M_TTCAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in **(4) Tx Event FIFO Element**.

The purpose of the Tx Event FIFO is to decouple handling transmit status information from transmit message handling i.e. a Tx Buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx Event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx Buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx Buffer before overwriting that Tx Buffer.

When a Tx Event FIFO full condition is signalled by MTTCAN0IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag MTTCAN0IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by MTTCAN0TXEFC.EFWM, interrupt flag MTTCAN0IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index MTTCAN0TXEFS.EFGI has to be added to the Tx Event FIFO start address MTTCAN0TXEFC.EFSA.

ISO CANFD ISO 11898-1:2015

#### 19.6.3.6 FIFO Acknowledge Handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see **(28) MTTCAN0RXF0A — Rx FIFO 0 Acknowledge**, **(32) MTTCAN0RXF1A — Rx FIFO 1 Acknowledge**, and **(46) MTTCAN0TXEFA — Tx Event FIFO Acknowledge**). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.

When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the M_TTCAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

**NOTE**

The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_TTCAN does not check for erroneous values.

ISO CANFD ISO 11898-1:2015

## 19.6.4 TTCAN Operation

### 19.6.4.1 Reference Message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the Time Master (sender of the reference message).

For Level 1 the data length must be at least one; for Level 0,2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to 8 potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register MTTCAN0TTRMC.

The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted Master_Ref_Mark is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (Next_is_Gap bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters.

The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

ISO CAN FD ISO 11898-1:2015

#### (1) Level 1

Level 1 operation is configured via MTTCAN0TTOCF.OM = “01” and MTTCAN0TTOCF.GEN. External clock synchronization is not available in Level 1.

The information related to the reference message is stored in the first data byte as shown in **Table 19.148** below. Cycle_Count is optional.

**Table 19.148 First byte of Level 1 reference message**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| First Byte | Next_is_Gap | res | Cycle_Count[5:0] | | | | | |

### (2) Level 2

Level 2 operation is configured via MTTCAN0TTOCF.OM = “10” and MTTCAN0TTOCF.GEN.

The information related to the reference message is stored in the first four data bytes as shown in **Table 19.149** below. Cycle_Count and the lower four bits of NTU_Res are optional. The M_TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

**Table 19.149 First four bytes of Level 2 reference message**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| First Byte | Next_is_Gap | res | Cycle_Count[5:0] | | | | | |
| Second Byte | NTU_Res[6:4] | | | NTU_Res[3:0] | | | | Disc_Bit |
| Third Byte | Master_Ref_Mark[7:0] | | | | | | | |
| Fourth Byte | Master_Ref_Mark[15:8] | | | | | | | |

### (3) Level 0

Level 0 operation is configured via MTTCAN0TTOCF.OM = “11”. External event-synchronized time-triggered operation is not available in Level 0.

The information related to the reference message is stored in the first four data bytes as shown in **Table 19.150** below. In Level 0 Next_is_Gap is always zero. Cycle_Count and the lower four bits of NTU_Res are optional. The M_TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

**Table 19.150 First four bytes of Level 0 reference message**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| First Byte | Next_is_Gap | res | Cycle_Count[5:0] | | | | | |
| Second Byte | NTU_Res[6:4] | | | NTU_Res[3:0] | | | | Disc_Bit |
| Third Byte | Master_Ref_Mark[7:0] | | | | | | | |
| Fourth Byte | Master_Ref_Mark[15:8] | | | | | | | |

ISO CANFD ISO 11898-1:2015

### 19.6.4.2 TTCAN Configuration

#### (1) TTCAN Timing

The Network Time Unit NTU is the unit in which all times are measured. The NTU is a constant of the whole network and is defined a priori by the network system designer. In TTCAN Level 1 the NTU is the nominal CAN bit time. In TTCAN Level 0 and Level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable.

In TTCAN Level 0 and Level 2 the length of the NTU is defined by the Time Unit Ratio TUR. The TUR is in principle a non-integer number and given by the formula TUR = MTTCAN0TURNA.NAV/ MTTCAN0TURCF.DC. The length of the NTU is given by the formula NTU = CAN Clock Period • TUR.

The TUR Numerator Configuration NC is an 18-bit number, MTTCAN0TURCF.NCL[15:0] can be programmed in the range $0000_H$ to $FFFF_H$. MTTCAN0TURCF.NCH[17:16] is hard wired to 0b01. When the number $nnnn_H$ is written to MTTCAN0TURCF.NCL[15:0], MTTCAN0TURNA.NAV starts with the value $10000_H + 0nnnn_H = 1nnnn_H$. The TUR Denominator Configuration MTTCAN0TURCF.DC is a 14-bit number. MTTCAN0TURCF.DC may be programmed in the range $0001_H$ to $3FFF_H$, $0000_H$ is an illegal value.

In Level 1, NC must be ≥ 4 • MTTCAN0TURCF.DC. In Level 0,2 NC must be ≥ 8 • MTTCAN0TURCF.DC to allow the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets MTTCAN0TURCF.DC to $1000_H$ and MTTCAN0TURCF.NCL to $10000_H$, resulting in an NTU consisting of 16 CAN clock periods. Local time and application watchdog are not started before either the MTTCAN0CCCR.INIT is reset, or MTTCAN0TURCF.ELT is set. MTTCAN0TURCF.ELT may not be set before the NTU is configured. Setting MTTCAN0TURCF.ELT to '1' also locks the write access to register MTTCAN0TURCF.

At startup MTTCAN0TURNA.NAV is updated from NC (= MTTCAN0TURCF.NCL + $10000_H$) when MTTCAN0TURCF.ELT is set. In TTCAN Level 1 there is no drift compensation. MTTCAN0TURNA.NAV does not change during operation, it always equals NC.

In TTCAN Level 0 and Level 2 there are two possibilities for MTTCAN0TURNA.NAV to change. When operating as time slave or backup time master, and when MTTCAN0TTOCF.ECC is set, MTTCAN0TURNA.NAV is updated automatically to the value calculated from the monitored global time speed, as long as the M_TTCAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization it returns to NC. When operating as the actual time master, and when MTTCAN0TTOCF.EECS is set, the Host may update MTTCAN0TURCF.NCL. When the Host sets MTTCAN0TTOCN.ECS, MTTCAN0TURNA.NAV will be updated from the new value of NC at the next reference message. The status flag MTTCAN0TTOST.WECS as is set when MTTCAN0TTOCN.ECS is set and is cleared when MTTCAN0TURNA.NAV is updated. MTTCAN0TURCF.NCL is write locked while MTTCAN0TTOST.WECS is set.

In TTCAN Level 0 and Level 2 the clock calibration process adapts MTTCAN0TURNA.NAV in the range of the Synchronization Deviation Limit SDL of NC ± $2^{(MTTCAN0TTOCF.LDSDL+5)}$. MTTCAN0TURCF.NCL should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of MTTCAN0TURNA.NAV.

The synchronization deviation SD is the difference between NC and MTTCAN0TURNA.NAV (SD = |NC - MTTCAN0TURNA.NAV|). It is limited by the Synchronization Deviation Limit SDL,

ISO CANFD ISO 11898-1:2015

which is configured by its dual logarithm MTTCAN0TTOCF.LDSDL (SDL = $2^{(MTTCAN0TTOCF.LDSDL+5)}$) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new Basic Cycle. When the calculated MTTCAN0TURNA.NAV deviates by more than SDL from NC, or if the Disc_Bit in the reference message is set, the drift compensation is suspended and MTTCAN0TTIR.GTE is set and TTOSC.QCS is reset, or in case of the Disc_Bit = '1', MTTCAN0TTIR.GTD is set.

TUR configuration examples are shown in **Table 19.151** below.

**Table 19.151 TUR Configuration Examples**

| TUR | 8 | 10 | 24 | 50 | 510 | 125000 | 32.5 | 100/12 | 529/17 |
|---|---|---|---|---|---|---|---|---|---|
| NC | $1FFF8_H$ | $1FFFE_H$ | $1FFF8_H$ | $1FFEA_H$ | $1FFFE_H$ | $1E848_H$ | $1FFE0_H$ | $19000_H$ | $10880_H$ |
| MTTCAN0TURCF.DC | $3FFF_H$ | $3333_H$ | $1555_H$ | $0A3D_H$ | $0101_H$ | $0001_H$ | $0FC0_H$ | $3000_H$ | $0880_H$ |

MTTCAN0TTOCN.ECS schedules NC for activation by the next reference message. MTTCAN0TTOCN.SGT schedules MTTCAN0TTGTP.TP for activation by the next reference message. Setting of MTTCAN0TTOCN.ECS and MTTCAN0TTOCN.SGT requires MTTCAN0TTOCF.EECS to be set (external clock synchronization enabled) while the M_TTCAN is actual time master.

The M_TTCAN module provides an application watchdog to verity the function of the application program. The Host has to serve this watchdog regularly, else all CAN bus activity is stopped. The Application Watchdog Limit MTTCAN0TTOCF.AWL specifies the number of NTUs between two times the watchdog has to be served. The maximum number of NTUs is 256. The Application Watchdog is served by reading register MTTCAN0TTOST. MTTCAN0TTOST.AWE indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, interrupt flag MTTCAN0TTIR.AW is set. For software development, the application watchdog may be disabled by programming MTTCAN0TTOCF.AWL to $00_H$ (see also **(10) Application Watchdog**).

ISO CAN FD ISO 11898-1:2015

(a) Timing of Interface Signals

The timing events which cause a pulse at output m_ttcan_tmp (MTTCAN0TMP) and m_ttcan_rtp (MTTCAN0RTP) are generated in the CAN clock domain. There is a clock domain crossing delay to be considered before the same event is visible in the Host clock domain (MTTCAN0TTIR.TTMI resp. MTTCAN0TTIR.RTMI set). The signals can be connected e.g. to the timing input(s) of another TTCAN node (m_ttcan_swt:MTTCAN0SWT / m_ttcan_evt: MTTCAN0EVT), in order to automatically synchronize two TTCAN networks.

Output m_ttcan_soc (MTTCAN0SOC) gets active whenever a reference message is completed (either transmitted or received). The output is controlled in the Host clock domain.

**(2) Message Scheduling**

MTTCAN0TTOCF.TM controls whether the M_TTCAN operates as a potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message's identifier MTTCAN0TTRMC.RID define the master priority, 0 giving the highest and 7 giving the lowest priority. There may not be two nodes in the network using the same master priority. MTTCAN0TTRMC.RID is used for recognition of reference messages. MTTCAN0TTRMC.RMPS is not relevant for time slaves.

The Initial Reference Trigger Offset MTTCAN0TTOCF.IRTO is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for

MTTCAN0TTOCF.IRTO is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, should correspond to their master priority, even with maximum clock drift.

MTTCAN0TTOCF.OM decides whether the node operates in TTCAN Level 0, Level 1, or Level 2. In one network, all potential time masters have to operate on the same level. Time slaves may operate on Level 1 in a Level 2 network, but not vice versa. The configuration of the TTCAN operation mode via MTTCAN0TTOCF.OM is the last step in the setup. With MTTCAN0TTOCF.OM = “00” (event-driven CAN communication), the M_TTCAN operates according to ISO 11898-1, without time triggers. With MTTCAN0TTOCF.OM = “01” (Level 1), the M_TTCAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the Next_is_Gap bit in the reference message is ignored. With MTTCAN0TTOCF.OM = “10” (Level 2), the M_TTCAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With MTTCAN0TTOCF.OM = “11” (Level 0), the M_TTCAN operates as event-driven CAN but maintains a calibrated global time base as in Level 2.

MTTCAN0TTOCF.EECS enables the external clock synchronization, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in Level 0,2 only) the global clock phase to an external reference.

MTTCAN0TTMLM.ENTT in the TT Matrix Limits register specifies the number of expected Tx_Triggers in the system matrix. This is the sum of Tx_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx_Ref_Triggers. Note that this is usually not the number of Tx_Trigger memory elements; the number of basic cycles in the system matrix and the trigger’s repeat factors have to be taken into account. An inaccurate configuration of MTTCAN0TTMLM.ENTT will result in either a Tx Count Underflow (MTTCAN0TTIR.TXU = ‘1’ and MTTCAN0TTOST.EL = “01”, severity 1) or in a Tx Count Overflow (MTTCAN0TTIR.TXO = ‘1’ and MTTCAN0TTOST.EL = “10”, severity 2).

**NOTE**

In case the first reference message seen by a node does not have Cycle_Count zero, this node may finish its first matrix cycle with its Tx count resulting in a Tx Count Underflow condition. As long as a node is in state Synchronizing its Tx_Triggers will not lead to transmissions.

MTTCAN0TTMLM.CCM specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of 8 basic cycles MTTCAN0TTMLM.CCM would be 7. MTTCAN0TTMLM.CCM is ignored by time slaves, a receiver of a reference message considers the received cycle count as the valid cycle count for the actual basic cycle.

MTTCAN0TTMLM.TXEW specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If the sample point of the first bit of a transmit message is not inside the Tx enable window because of e.g. a slight overlap from the previous time window’s message, the transmission cannot be started in that time window at all. MTTCAN0TTMLM.TXEW has to be chosen with respect to the network’s synchronization quality and with respect to the relation between the length of the time windows and the length of the messages.

ISO CAN FD ISO 11898-1:2015

### (3) Trigger Memory

The trigger memory is part of the external Message RAM to which the M_TTCAN is connected via its Generic Master Interface (see **Figure 19.16, Message RAM Configuration**). It stores up to 64 trigger elements. A trigger memory element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, Filter Type FTYPE, Message Number MNR, Message Status Count MSC, Time Mark Event Internal TMIN, Time Mark Event External TMEX, and Asynchronous Serial Communication ASC (see **(7) Trigger Memory Element**).

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory have to be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are ignored for triggers of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, and End_of_List.

When the cycle time reaches the time mark of the actual trigger, the FSE switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a transmit trigger, the Tx Handler starts to read the message from the Message RAM as soon as the FSE switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx Handler has to be able to prepare the transmission before the transmit trigger's time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mark is reached. If the reference message is n NTU long, a trigger with a time mark< n will never become active and will be treated as a configuration error.

Starting point of the cycle time is the sample point of the reference message's start of frame bit. The next reference message is requested when cycle time reaches the Tx_Ref_Trigger's time mark. The M_TTCAN reacts on the transmission request at the next sample point. A new Sync_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync_Mark is taken as the new Ref_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialisation) be seen at a value< n, with n being the length of the reference message measured in NTU.

Length of a basic cycle: Tx_Ref_Trigger's time mark+1 NTU+1 CAN bit time

The trigger list will be different for all nodes in the TTCAN network. Each node knows only the Tx_Triggers for its own transmit messages, the Rx_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

#### (a) Trigger Types

Tx_Ref_Trigger (TYPE = "0000") and Tx_Ref_Trigger_Gap (TYPE = "0001") cause the transmission of a reference message by a time master. A configuration error (MTTCAN0TTOST.EL = "11", severity 3) is detected when a time slave encounters a Tx_Ref_Trigger(_Gap) in its trigger memory. Tx_Ref_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, Tx_Ref_Trigger is ignored when the M_TTCAN synchronization state is In_Gap (MTTCAN0TTOST.SYS = "10").

Tx_Trigger_Single (TYPE = "0010"), Tx_Trigger_Continuous (TYPE = "0011"), Tx_Trigger_Arbitration (TYPE = "0100"), and Tx_Trigger_Merged (TYPE = "0101") cause the start of a transmission. They define the start of a time window.

Tx_Trigger_Single starts a single transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit is reset.

ISO CANFD ISO 11898-1:2015

Tx_Trigger_Continuous starts a transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit remains set, and the message buffer is transmitted again in the next matching time window.

Tx_Trigger_Arbitration starts an arbitrating time window, Tx_Trigger_Merged a merged arbitrating time window. The last Tx_Trigger of a merged arbitrating time window must be of type Tx_Trigger_Arbitration. A Configuration Error (MTTCAN0TTOST.EL = "11", severity 3) is detected when a trigger of type Tx_Trigger_Merged is followed by any other Tx_Trigger than one of type Tx_Trigger_Merged or Tx_Trigger_Arbitration. Several Tx_Triggers may be defined for the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code has to be considered when the expected number of Tx_Triggers (MTTCAN0TTMLM.ENTT) is calculated.

Watch_Trigger (TYPE = "0110") and Watch_Trigger_Gap (TYPE = "0111") check for missing reference messages. They are used by both time masters and time slaves. Watch_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch_Trigger is ignored when the M_TTCAN synchronization state is In_Gap (MTTCAN0TTOST.SYS = "10").

Rx_Trigger (TYPE = "1000") is used to check for the reception of periodic messages in exclusive time windows. Rx_Triggers are not active until state In_Schedule or In_Gap is reached. The time mark of an Rx_Trigger shall be placed after the end of that message's transmission, independent of time window boundaries. Depending on their cycle code, Rx_Triggers may be ignored in some basic cycles. At the time mark of the Rx_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx_Trigger had matched the acceptance filter element referenced by MNR. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx_Trigger. Acceptance filter elements which are referenced by Rx_Triggers should be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx_Trigger's time mark is reached.

Time_Base_Trigger (TYPE = "1001") are used to generate internal/external events depending on the configuration of ASC, TMIN, and TMEX.

End_of_List (TYPE = "1010 to 1111") is an illegal trigger type, a configuration error (MTTCAN0TTOST.EL = "11", severity 3) is detected when an End_of_List trigger is encountered in the trigger memory before the Watch_Trigger or Watch_Trigger_Gap.

ISO CANFD ISO 11898-1:2015

(b) Restrictions for the Node's Trigger List

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark.

Rx_Triggers and Time_Base_Triggers may not be placed inside the Tx enable windows of Tx_Trigger_Single/Continuous/Arbitration, but they may be placed after Tx_Trigger_Merged.

Triggers that are placed after the Watch_Trigger (or the Watch_Trigger_Gap when MTTCAN0TTOST.SYS = "10") will never become active. The watch triggers themselves will not become active when the reference messages are transmitted on time.

All unused trigger memory words (after the Watch_Trigger or after the Watch_Trigger_Gap when MTTCAN0TTOST.SYS = "10") must be set to trigger type End_of_List.

A typical trigger list for a potential time master will begin with a number of Tx_Triggers and Rx_Triggers followed by the Tx_Ref_Trigger and the Watch_Trigger. For networks with external event- synchronized time-triggered communication, this is followed by the Tx_Ref_Trigger_Gap and the Watch_Trigger_Gap. The trigger list for a time slave will be the same but without the Tx_Ref_Trigger and the Tx_Ref_Trigger_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger's time mark and activates the trigger. Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx_Ref_Trigger and Tx_Ref_Trigger_Gap. In a time master competing for master ship, the effective time mark of a Tx_Ref_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx_Ref_Trigger or Tx_Ref_Trigger_Gap is the sum of its configured time mark and the Reference Trigger Offset MTTCAN0TTOCF.IRTO. In case error level 2 is reached (MTTCAN0TTOST.EL = "10"), the effective time mark is the sum of its time mark and $127_H$. No other trigger elements should be placed in this range otherwise it may happen, that the time marks appear out of order and are flagged as a configuration error. Trigger elements which are coming after Tx_Ref_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:

- Host clock frequency
- Speed and waiting time for Trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

### (c) Example for Trigger Handling

The example below shows how the trigger list is derived from a node's system matrix. Assumed node A is first time master and has knowledge of the section of the system matrix shown in **Table 19.152** below.

ISO CANFD ISO 11898-1:2015

**Table 19.152 System Matrix Node A**

| Cycle Count | Time Mark1 | Time Mark2 | Time Mark3 | Time Mark4 | Time Mark5 | Time Mark6 | Time Mark7 |
|---|---|---|---|---|---|---|---|
| 0 | Tx7 | | | | | TxRef | Error |
| 1 | Rx3 | | | Tx2, Tx4 | | TxRef | Error |
| 2 | | | | | | TxRef | Error |
| 3 | Tx7 | | Rx5 | | | TxRef | Error |
| 4 | Tx7 | | | Rx6 | | TxRef | Error |

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the number of basic cycles in the system matrix is 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by MTTCAN0TTMLM.CCM. The Cycle Code CC is composed of repeat factor (= value of most significant '1') and the number of the first basic cycle in the system matrix (= bit field after most significant '1').

Example:with a cycle code of 0b0010011 (repeat factor: 16, first basic cycle: 3) and a maximum cycle count of MTTCAN0TTMLM.CCM = "$3F_H$"matches occur at cycle counts 3, 19, 35, 51

A trigger element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, and Message Number MNR. For transmission MNR references the Tx Buffer number (0 to 31). For reception MNR references the number of the filter element (0 to 127) that matched during acceptance filtering.

Depending on the configuration of the Filter Type FTYPE, the 11-bit or 29-bit message ID filter list is referenced.

In addition a trigger element can be configured for Asynchronous Serial Communication ASC, generation of Time Mark Event Internal TMIN, and Time Mark Event External TMEX. The Message Status Count MSC holds the counter value (0 to 7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

**Table 19.153 Trigger List Node A**

| Trigger | Time Mark TM[15:0] | Cycle Code CC[6:0] | Trigger Type TYPE[3:0] | Mess. No. MNR[6:0] |
|---|---|---|---|---|
| 0 | Mark1 | $0000100_B$ | Tx_Trigger_Single | 7 |
| 1 | Mark1 | $1000000_B$ | Rx_Trigger | 3 |
| 2 | Mark1 | $1000011_B$ | Tx_Trigger_Single | 7 |
| 3 | Mark3 | $1000001_B$ | Tx_Trigger_Merged | 2 |
| 4 | Mark3 | $1000011_B$ | Rx_Trigger | 5 |
| 5 | Mark4 | $1000001_B$ | Tx_Trigger_Arbitration | 4 |
| 6 | Mark4 | $1000100_B$ | Rx_Trigger | 6 |
| 7 | Mark6 | n.a. | Tx_Ref_Trigger | 0 (Ref) |
| 8 | Mark7 | n.a. | Watch_Trigger | n.a. |
| 9 | n.a. | n.a. | End_of_List | n.a. |

Tx_Trigger_Single, Tx_Trigger_Continuous, Tx_Trigger_Merged, Tx_Trigger_Arbitration, Rx_Trigger, and Time_Base_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored.

The FSE starts the basic cycle with scanning the trigger list starting from zero until a trigger with time mark > cycle time and with its Cycle Code CC matching the actual cycle count is reached, or a trigger of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, or Watch_Trigger_Gap is encountered.

When the cycle time reached the Time Mark TM, the action defined by Trigger Type TYPE and Message Number MNR is started. There is an error in the configuration when End_of_List is reached.

At Mark6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the Watch Trigger at Mark7 is reached, the node was not able to transmit the reference message; error treatment is started.

ISO CAN FD ISO 11898-1:2015

### (d) Detection of Configuration Errors

A configuration error is signalled via MTTCAN0TTOST.EL = “11” (severity 3) when:

The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time.

The previous active trigger was a Tx_Trigger_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a Tx_Trigger_Merged nor a Tx_Trigger_Arbitration nor a Time_Base_Trigger nor an Rx_Trigger.

The FSE of a node with MTTCAN0TTOCF.TM=‘0’ (time slave) encounters a Tx_Ref_Trigger or a Tx_Ref_Trigger_Gap.

Any time mark placed inside the Tx enable window (defined by MTTCAN0TTMLM.TXEW) of a Tx_Trigger with a matching cycle code.

A time mark is placed near the time mark of a Tx_Ref_Trigger and the Reference Trigger Offset MTTCAN0TTOST.RTO causes a reversal of their sequential order measured in cycle time.

### (4) TTCAN Schedule Initialization

The synchronization to the M_TTCAN's message schedule starts when MTTCAN0CCCR.INIT is reset. The M_TTCAN can operate strictly time-triggered (MTTCAN0TTOCF.GEN = '0') or external event-synchronized time-triggered (MTTCAN0TTOCF.GEN = '1'). All nodes start with cycle time zero at the beginning of their trigger list with MTTCAN0TTOST.SYS = "00" (out of synchronization), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronized time-triggered operation mode will ignore Tx_Ref_Trigger and Watch_Trigger and will use instead Tx_Ref_Trigger_Gap and Watch_Trigger_Gap until the first reference message decides whether a Gap is active.

#### (a) Time Slaves

After configuration, a time slave will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message before reaching the Watch_Triggers. When it reaches Init_Watch_Trigger, interrupt flag MTTCAN0TTIR.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). The first received reference message will restart the FSE and the cycle time.

**NOTE**

Init_Watch_Trigger is not part of the trigger list. It is implemented as an internal counter which counts up to $FFFF_H$ = maximum cycle time.

When a time slave has received any message but the reference message before reaching the Watch_Triggers, it will assume a fatal error (MTTCAN0TTOST.EL = "11", severity 3), set interrupt flag MTTCAN0TTIR.WT, switch off its CAN bus output, and enter the bus monitoring mode (MTTCAN0CCCR.MON set to '1'). In the bus monitoring mode it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

**NOTE**

To leave the fatal error state, the Host has to set MTTCAN0CCCR.INIT = '1'. After reset of MTTCAN0CCCR.INIT, the node restarts TTCAN communication.

When no error is encountered during synchronization, the first reference message sets MTTCAN0TTOST.SYS = "01" (Synchronizing), the second sets the TTCAN synchronization state (depending on its Next_is_Gap bit) to MTTCAN0TTOST.SYS = "11" (In_Schedule) or MTTCAN0TTOST.SYS = "10" (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

#### (b) Potential Time Masters

After configuration, a potential time master will start the transmission of a reference message when it reaches its Tx_Ref_Trigger (or its Tx_Ref_Trigger_Gap when in external event-synchronized time-triggered operation). It will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message or transmit the reference message successfully before reaching the Watch_Triggers (assumed reason: all other nodes still in reset or configuration, giving no acknowledge). When it reaches Init_Watch_Trigger, the attempted transmission is aborted, interrupt flag MTTCAN0TTIR.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting MTTCAN0TTIR.IWT will re-enable the transmission of reference messages until next time the Init_Watch_Trigger condition is met, or another CAN message is received. The FSE will be restarted by the reception of a reference message.

ISO Draft NFD ISO 11898-1:2015

When a potential time master reaches the Watch_Triggers after it has received any message but the reference message, it will assume a fatal error (MTTCAN0TTOST.EL = “11”, severity 3), set interrupt flag MTTCAN0TTIR.WT, switch off its CAN bus output, and enter the bus monitoring mode (MTTCAN0CCCR.MON set to ‘1’). In bus monitoring mode, it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets MTTCAN0TTOST.SYS = “01” (synchronizing), the second sets the TTCAN synchronization state (depending on its Next_is_Gap bit) to MTTCAN0TTOST.SYS = “11” (In_Schedule) or MTTCAN0TTOST.SYS = “10” (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

A potential time master is current time master (MTTCAN0TTOST.MS = “11”) when it was the transmitter of the last reference message, else it is backup time master (MTTCAN0TTOST.MS = “10”).

When all potential time masters have finished configuration, the node with the highest time master priority in the network will become the current time master.

### 19.6.4.3 TTCAN Gap Control

All functions related to Gap control apply only when the M_TTCAN is operated in external event-synchronized time-triggered mode (MTTCAN0TTOCF.GEN = ‘1’). In this operation mode the TTCAN message schedule may be interrupted by inserting Gaps between the basic cycles of the system matrix. All nodes connected to the CAN network have to be configured for external event-synchronized time-triggered operation.

During a Gap, all transmissions are stopped and the CAN bus remains idle. A Gap is finished when the next reference message starts a new basic cycle. A Gap starts at the end of a basic cycle that itself was started by a reference message with bit Next_is_Gap = ‘1’ e.g. Gaps are initiated by the current time master.

The current time master has two options to initiate a Gap. A Gap can be initiated under software control when the application program writes MTTCAN0TTOCN.NIG = ‘1’. The Next_is_Gap bit will be transmitted as ‘1’ with the next reference message. A Gap can also be initiated under hardware control when the application program enables the event trigger input pin m_ttcan_evt (MTTCAN0EVT) by writing MTTCAN0TTOCN.GCS = ‘1’. When a reference message is started and MTTCAN0TTOCN.GCS is set, a HIGH level at pin m_ttcan_evt (MTTCAN0EVT) will set Next_is_Gap = ‘1’.

As soon as that reference message is completed, the MTTCAN0TTOST.WFE bit will announce the Gap to the time master as well as to the time slaves. The current basic cycle will continue until its last time window. The time after the last time window is the Gap time.

For the actual time master and the potential time masters, MTTCAN0TTOST.GSI will be set when the last basic cycle has finished and the Gap time starts. In nodes that are time slaves, bit MTTCAN0TTOST.GSI will remain at ‘0’.

When a potential time master is in synchronization state In_Gap (MTTCAN0TTOST.SYS = “10”), it has four options to intentionally finish a Gap:

Under software control by writing MTTCAN0TTOCN.FGP = ‘1’.

Under hardware control (MTTCAN0TTOCN.GCS = ‘1’) an edge from HIGH to LOW at the event-trigger input pin m_ttcan_evt (MTTCAN0EVT) sets MTTCAN0TTOCN.FGP and restarts the schedule.

ISO 11898-1:2015

The third option is a time-triggered restart. When MTTCAN0TTOCN.TMG = '1', the next register time mark interrupt (MTTCAN0TTIR.RTMI = '1') will set MTTCAN0TTOCN.FGP and start the reference message.

Finally any potential time master will finish a Gap when it reaches its Tx_Ref_Trigger_Gap, assuming that the event to synchronize on did not occur in time.

Neither of these options can cause a basic cycle to be interrupted with a reference message.

Setting of MTTCAN0TTOCN.FGP after the Gap time has started will start the transmission of a reference message immediately and will thereby synchronize the message schedule. When MTTCAN0TTOCN.FGP is set before the Gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the Tx_Ref_Trigger – there will be no Gap time in the message schedule.

In strictly time-triggered operation, bit Next_is_Gap = '1' in the reference message will be ignored, as well as the event-trigger input pin m_ttcan_evt (MTTCAN0EVT) and the bits MTTCAN0TTOCN.NIG, MTTCAN0TTOCN.FGP, and MTTCAN0TTOCN.TMG.

#### 19.6.4.4 Stop Watch

The stop watch function enables capturing of M_TTCAN internal time values (local time, cycle time, or global time) triggered by an external event.

To enable the stop watch function, the application program first has to define local time, cycle time, or global time as stop watch source via MTTCAN0TTOCN.SWS. When MTTCAN0TTOCN.SWS is ≠ "00" and TT Interrupt Register flag MTTCAN0TTIR.SWE is '0', the actual value of the time selected by MTTCAN0TTOCN.SWS will be copied into MTTCAN0TTCPT.SWV on the next rising/falling edge (as configured via MTTCAN0TTOCN.SWP) on pin m_ttcan_swt (MTTCAN0SWT). This will set interrupt flag MTTCAN0TTIR.SWE. After the application program has read MTTCAN0TTCPT.SWV, it may enable the next stop watch event by resetting MTTCAN0TTIR.SWE to '0'.

#### 19.6.4.5 Local Time, Cycle Time, Global Time, and External Clock Synchronization

There are two possible levels in time-triggered CAN: Level 1 and Level 2. Level 1 only provides time-triggered operation using cycle time. Level 2 additionally provides increased synchronization quality, global time and external clock synchronization. In both levels, all timing features are based on a local time base - the local time.

The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented 8 times each NTU. If the length of the NTU is shorter than 8 CAN clock periods (as may be configured in Level 1, or as a result of clock calibration in Level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only 4 times each NTU.

**Figure 19.27** describes the synchronization of the cycle time and global time, performed in the same manner by all TTCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message's frame synchronization event. This frame synchronization event occurs at the sample point of each Start of Frame (SoF) bit and causes the local time to be stored as Sync_Mark. Sync_Marks and Ref_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the Cycle Sync Mark

ISO CANFD ISO 11898-1:2015

(Cycle Sync Mark = Sync_Mark - Ref_Mark) stored in register MTTCAN0TTCSM. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (Cycle Time = Local Time - Ref_Mark).

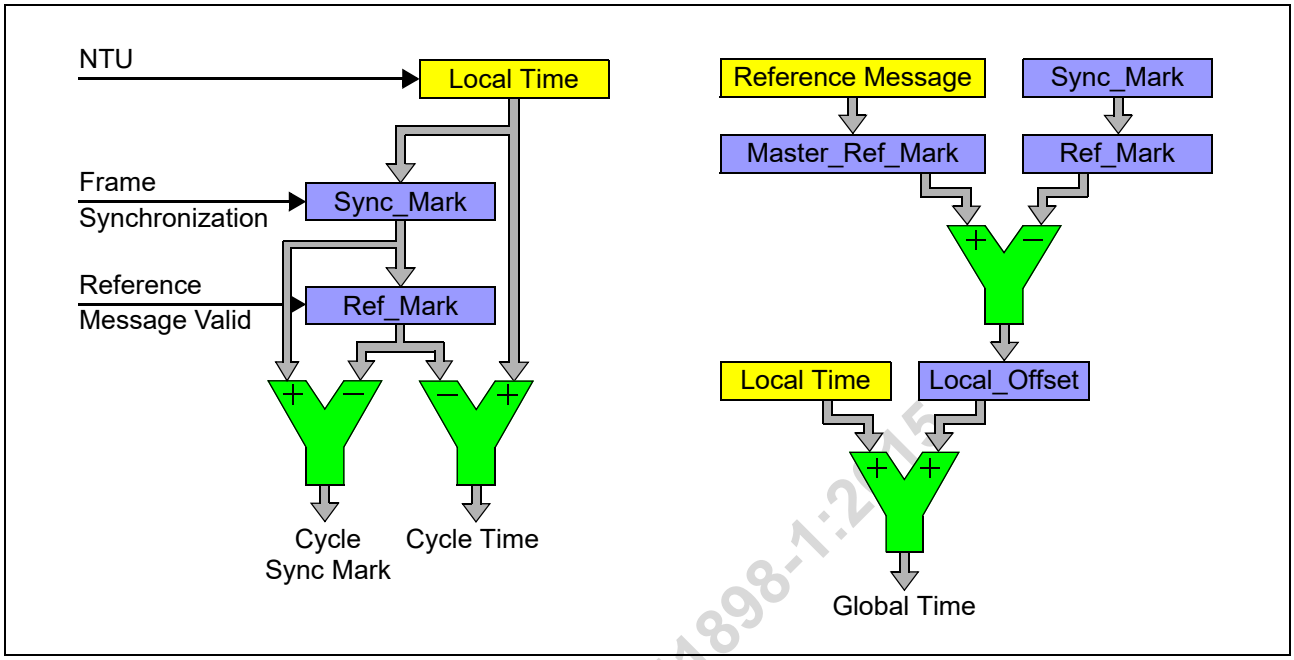



**Figure 19.27 Cycle Time and Global Time Synchronization**

ISO CAN FD ISO 11898-1:2015

The cycle time that can be read from MTTCAN0TTCTC.CT is the difference of the node's local time and Ref_Mark, both synchronized into the Host clock domain and truncated to 16 bit.

The global time exists for TTCAN Level 0 and Level 2 only, in Level 1 it is invalid. The node's view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master will use its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3,4). The global time that can be read from MTTCAN0TTLGT.GT is the sum of the node's local time and its local offset, both synchronized into the Host clock domain and truncated to 16 bit. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark (see **Figure 19.27**). The node's view of the global time is local time + local offset. In a potential time master that has never received another time master's reference message, Local_Offset will be zero. When a node becomes the current time master after first having received other reference messages, Local_Offset will be frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signalled by Disc_Bit in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register MTTCAN0TTLGT is smoothed by a low-pass filtering to have a continuous monotonic value.

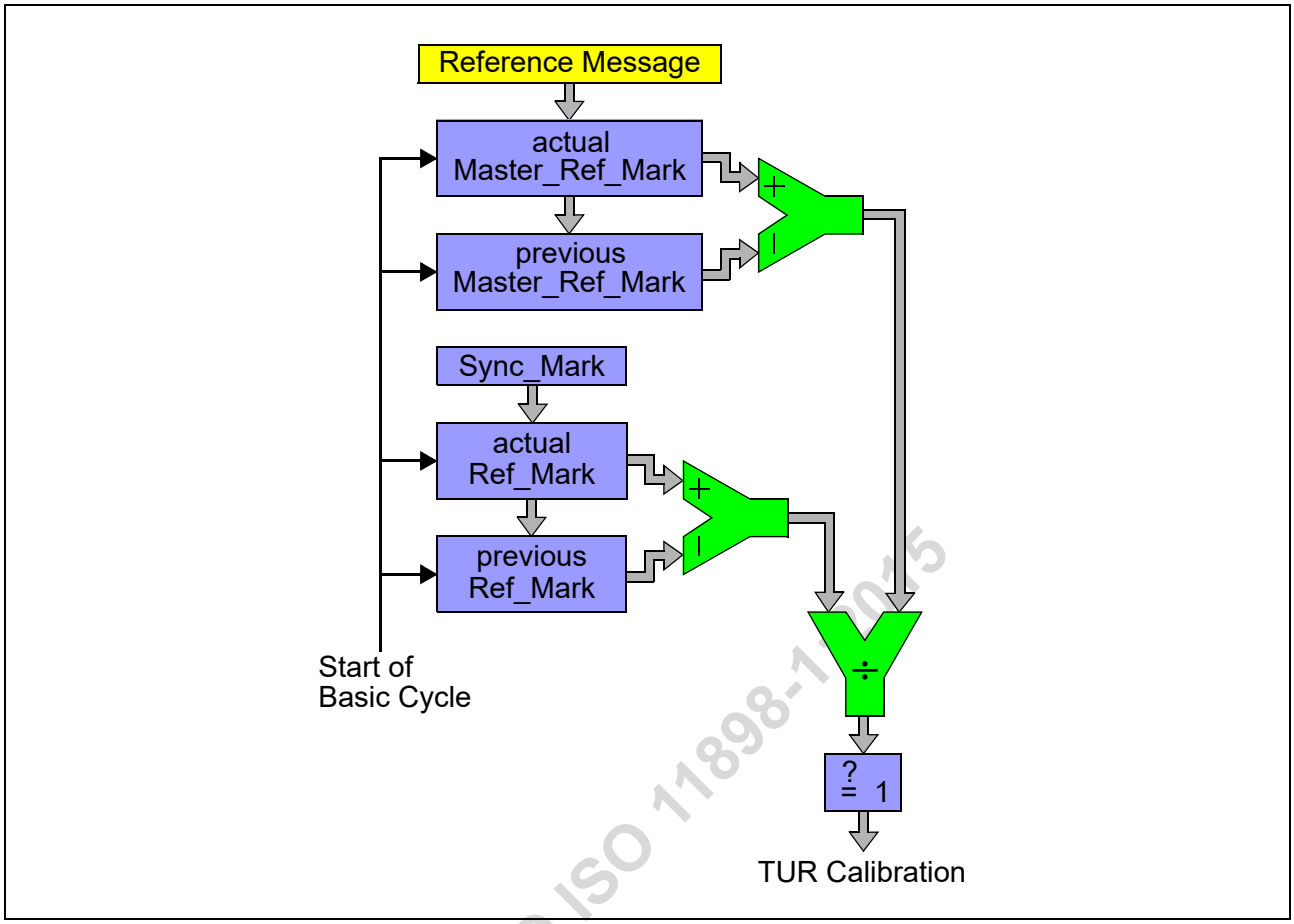



Figure 19.28 TTCAN Level 0 and Level 2 Drift Compensation

**Figure 19.28** describes how in TTCAN Level 0,2 each time receiving node compensates the drift between its own local clock and the time master's clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the Disc_Bit in the reference message is not set, a new value for MTTCAN0TURNA.NAV is calculated. If the Synchronization Deviation SD = |NC - MTTCAN0TURNA.NAV| ≤ SDL (Synchronization Deviation Limit), the new value for MTTCAN0TURNA.NAV takes effect. Else the automatic drift compensation is suspended.

In TTCAN Level 0 and Level 2, MTTCAN0TTOST.QCS indicates whether the automatic drift compensation is active or suspended. In TTCAN Level 1, MTTCAN0TTOST.QCS is always '1'.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit MTTCAN0TTOCF.EECS.

The stop watch function (see **Section 19.6.4.4, Stop Watch**) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated Numerator Configuration Low to MTTCAN0TURCF.NCL (MTTCAN0TURCF.DC cannot be updated during operation). The new value takes effect by writing MTTCAN0TTOCN.ECS to '1'.

The global time phase is adjusted by first writing the phase offset into the TT Global Time Preset register MTTCAN0TTGTP. The new value takes effect by writing MTTCAN0TTOCN.SGT to '1'. The first reference message transmitted after the global time phase adjustment will have the Disc_Bit set to '1'.

ISO CANFD ISO 11898-1:2015

MTTCAN0TTOST.QGTP shows whether the node's global time is in phase with the time master's global time. MTTCAN0TTOST.QGTP is permanently '0' in TTCAN Level 1 and when the Synchronization Deviation Limit is exceeded in TTCAN Level 0,2 (MTTCAN0TTOST.QCS = '0'). It is temporarily '0' while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a Disc_Bit = '1' or when MTTCAN0TTOST.QCS = '0'.

### 19.6.4.6 TTCAN Error Level

The ISO 11898-4 specifies four levels of error severity:

#### S0 - No Error

#### S1 - Warning

Only notification of application, reaction application-specific.

#### S2 Error

Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the Reference Trigger Offset MTTCAN0TTOST.RTO set to the maximum value of 127.

#### S3 - Severe Error

Notification of application. All CAN bus operations are stopped, i.e. transmission of dominant bits is not allowed, and MTTCAN0CCCR.MON is set. The S3 error condition remains active until the application updates the configuration (set MTTCAN0CCCR.CCE).

If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by MTTCAN0TTIR.ELC. The error level is monitored by MTTCAN0TTOST.EL.

The M_TTCAN signals the following error conditions as required by ISO 11898-4:

#### Config_Error (S3)

Sets Error Level MTTCAN0TTOST.EL to "11" when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.

#### Watch_Trigger_Reached (S3)

Sets Error Level MTTCAN0TTOST.EL to "11" when a watch trigger was reached because the reference message is missing.

#### Application_Watchdog (S3)

Sets Error Level MTTCAN0TTOST.EL to "11" when the application failed to serve the application watchdog. The application watchdog is configured via MTTCAN0TTOCF.AWL. It is served by reading register MTTCAN0TTOST. When the watchdog is not served in time, bit MTTCAN0TTOST.AWE and interrupt flag MTTCAN0TTIR.AW are set, all TTCAN communication is stopped, and the M_TTCAN is set into bus monitoring mode (MTTCAN0CCCR.MON set to '1').

ISO CAN FD ISO 11898-1:2015

#### CAN_Bus_Off (S3)

Entering CAN_Bus_Off state sets error level MTTCAN0TTOST.EL to “11”. CAN_Bus_Off state is signalled by MTTCAN0PSR.BO = ‘1’ and MTTCAN0CCCR.INIT = ‘1’.

#### Scheduling_Error_2 (S2)

Sets Error Level MTTCAN0TTOST.EL to “10” if the MSC of one Tx_Trigger has reached 7. In addition interrupt flag MTTCAN0TTIR.SE2 is set. The Error Level MTTCAN0TTOST.EL is reset to “00” at the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.

#### Tx_Overflow (S2)

Sets Error Level MTTCAN0TTOST.EL to “10” when the Tx count is equal or higher than the expected number of Tx_Triggers MTTCAN0TTMLM.ENTT and a Tx_Trigger event occurs. In addition interrupt flag MTTCAN0TTIR.TXO is set. The Error Level MTTCAN0TTOST.EL is reset to “00” when the Tx count is no more than MTTCAN0TTMLM.ENTT at the start of a new matrix cycle.

#### Scheduling_Error_1 (S1)

Sets Error Level MTTCAN0TTOST.EL to “01” if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than 2, or if one of the MSCs of an exclusive Rx_Trigger has reached 7. In addition interrupt flag MTTCAN0TTIR.SE1 is set. If within one matrix cycle none of these conditions is valid, the Error Level MTTCAN0TTOST.EL is reset to “00”.

#### Tx_Underflow (S1)

Sets Error Level MTTCAN0TTOST.EL to “01” when the Tx count is less than the expected number of Tx_Triggers MTTCAN0TTMLM.ENTT at the start of a new matrix cycle. In addition interrupt flag MTTCAN0TTIR.TXU is set. The Error Level MTTCAN0TTOST.EL is reset to “00” when the Tx count is at least MTTCAN0TTMLM.ENTT at the start of a new matrix cycle.

ISO CANFD ISO 11898-1:2015

### 19.6.4.7 TTCAN Message Handling

#### (1) Reference Message

For potential time masters the identifier of the reference message is configured via MTTCAN0TTRMC.RID. No dedicated Tx Buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte (TTCAN Level 1) resp. the first four data bytes (TTCAN Level 0 and Level 2) will be provided by the FSE.

In case the reference message Payload Select MTTCAN0TTRMC.RMPS is set, the rest of the reference message's payload (Level 1: bytes 2-8, Level 0,2: bytes 5-6) is taken from Tx Buffer 0. In this case the data length DLC code from message buffer 0 is used.

**Table 19.154 Number of Data Bytes transmitted with a reference messages**

| MTTCAN0TTRMC. RMPS | MTTCAN0TXBRP. TRP0 | Level 0 | Level 1 | Level 2 |
|---|---|---|---|---|
| 0 | 0 | 4 | 1 | 4 |
| 0 | 1 | 4 | 1 | 4 |
| 1 | 0 | 4 | 1 | 4 |
| 1 | 1 | 4 + MB0 | 1 + MB0 | 4 + MB0 |

To send additional payload with the reference message in Level 1 a DLC > 1 has to be configured, for Level 0,2 a DLC > 4 is required. In addition the transmission request pending bit MTTCAN0TXBRP.TRP0 of message buffer 0 must be set (see **Table 19.154**). In case bit MTTCAN0TXBRP.TRP0 is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the Reference Identifier MTTCAN0TTRMC.RID is used.

#### (2) Message Reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see **Section 19.6.3.4, Rx Handling**).

The Message Status Count MSC is part of the corresponding trigger memory element and has to be initialized to zero during configuration. It is updated while the M_TTCAN is in synchronization states In_Gap or In_Schedule. The update happens at the message's Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has completed. This has to take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

ISO CANFD ISO 11898-1:2015

### (3) Message Transmission

For time-triggered message transmission the M_TTCAN supplies 32 dedicated Tx buffers (see **Section (2), Dedicated Tx Buffers**). A Tx FIFO or Tx queue is not available when the M_TTCAN is configured for time-triggered operation (MTTCAN0TTOCF.OM = “01” or “10”).

Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle.

The application program has to update the data regularly and on time, synchronized to the cycle time. The Host CPU is responsible that no partially updated messages are transmitted. To assure this the Host has to proceed in the following way:

Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration

- Check whether the previous transmission has completed by reading MTTCAN0TXBTO
- Update the Tx buffer’s configuration and/or payload
- Issue an Add Request to set the Tx Buffer Request Pending bit

Tx_Trigger_Continuous

- Issue a Cancellation Request to reset the Tx Buffer Request Pending bit
- Check whether the cancellation has finished by reading MTTCAN0TXBCF
- Update Tx buffer’s configuration and/or payload
- Issue an Add Request to set the Tx Buffer Request Pending bit

The message’s MSC stored with the corresponding Tx_Trigger provides information on the success of the transmission.

The MSC is incremented by one when the transmission could not be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and could not be completed successfully. The MSC is decremented by one when the message was transmitted successfully or when the message could have been started within its transmit enable window but was not started because transmission was disabled (M_TTCAN in Error Level S2 or Host has disabled this particular message).

The Tx buffers may be managed dynamically, i.e. several messages with different identifiers may share the same Tx buffer element. In this case the Host has to assure that no transmission request is pending for the Tx buffer element to be reconfigured by checking MTTCAN0TXBRP.

If a Tx buffer with pending transmission request should be updated, the Host first has to issue a cancellation request and check whether the cancellation has completed by reading MTTCAN0TXBCF before it starts updating.

The Tx Handler will transfer a message from the Message RAM to its intermediate output buffer at the trigger element which becomes active immediately before the Tx_Trigger element which defines the beginning of the transmit window. During and after the transfer time the transmit message may not be updated and its MTTCAN0TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be e.g. a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger has to be large enough to guarantee that the Tx Handler can read four words from the Message RAM even at high RAM access load from other modules.

ISO CANFD ISO 11898-1:2015

(a) Transmission in Exclusive Time Windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continuous. There is no arbitration on the bus with messages from other nodes. The MSC is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx Buffer Request Pending bit is reset. After successful transmission started by a Tx_Trigger_Continuous the respective Tx Buffer Request Pending remains set. When the transmission was not successful due to disturbances, it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

(b) Transmission in Arbitrating Time Windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message has to arbitrate with the messages from other nodes. The MSC is not updated. When the transmission was not successful (lost arbitration or disturbance), it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

(c) Transmission in Merged Arbitrating Time Windows

The purpose of a merged arbitrating time window is to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a single node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission will not be started if the corresponding Transmission Request Pending flag was reset by a successful Tx cancellation.

In single transmit windows, the Tx Handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmissions will be attempted in the sequence defined by the trigger list. If the time mark of a fourth message is read before the first is transmitted (or cancelled by the Host), the fourth request will be ignored.

The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window will be started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if there is competing bus traffic. This has to be considered for the configuration of the trigger list. Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx Buffer needs to be updated.

ISO CAN FD ISO 11898-1:2015

#### 19.6.4.8 TTCAN Interrupt and Error Handling

The TT Interrupt Register MTTCAN0TTIR consists of four segments. Each interrupt can be enabled separately by the corresponding bit in the TT Interrupt Enable register MTTCAN0TTIE. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position.

The first segment consists of flags CER, AW, WT, and IWT. Each flag indicates a fatal error condition where the CAN communication is stopped. With the exception of IWT, these error conditions require a re-configuration of the M_TTCAN module before the communication can be restarted.

The second segment consists of flags ELC, SE1, SE2, TXO, TXU, and GTE. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, e.g. by disturbances on the CAN bus, they will be handled by the TTCAN protocol's failure handling and do not require intervention by the application program.

The third segment consists of flags GTD, GTW, SWE, TTMI, and RTMI. The first two flags are controlled by global time events (Level 0, 2 only) that require a reaction by the application program. With a Stop Watch Event triggered by a rising/falling edge on pin m_ttcan_swt (MTTCAN0SWT) internal time values are captured. The Trigger Time Mark Interrupt notifies the application that a specific Time_Base_Trigger is reached. The Register Time Mark Interrupt signals that the time referenced by MTTCAN0TTOCN.TMC (Cycle, Local, or Global) equals time mark MTTCAN0TTTMK.TM. It can also be used to finish a Gap.

The fourth segment consists of flags SOG, CSM, SMC, and SBC. These flags provide a means to synchronize the application program to the communication schedule.

ISO CANFD ISO 11898-1:2015

#### 19.6.4.9 Level 0

TTCAN Level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in TTCAN Level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1:2015.

Level 0 operation is configured via MTTCAN0TTOCF.OM = “11”. In this mode the M_TTCAN operates in event-driven CAN communication, there is no fixed schedule, the configuration of MTTCAN0TTOCF.GEN is ignored. External event-synchronized operation is not available in Level 0. A synchronized time base is maintained by transmission of reference messages.

In Level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (MTTCAN0TTIR.TTMI) is set when the cycle time has reached MTTCAN0TTOCF.IRTO • $200_H$, it reminds the Host to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (MTTCAN0TTIR.WT) is set when the cycle time has reached $FF00_H$. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks.

Register time mark interrupts (MTTCAN0TTIR.RTMI) are also possible.

The reference message is configured as for Level 2 operation. Received reference messages are recognized by the identifier configured in register MTTCAN0TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the Host sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- MTTCAN0TTRMC
- MTTCAN0TTOCF except EVTP, AWL, GEN
- MTTCAN0TTMLM except ENTT, TXEW
- MTTCAN0TURCF

Level 0 operation is controlled via:

- MTTCAN0TTOCN except NIG, TMG, FGP, GCS, TTMIE
- MTTCAN0TTGTP
- MTTCAN0TTTMK
- MTTCAN0TTIR excluding bits CER, AW, IWT SE2, SE1, TXO, TXU, SOG (no function)
- MTTCAN0TTIR the following bits have changed function
  - TTMI not defined by trigger memory - activated at cycle time MTTCAN0TTOCF.IRTO • $200_H$
  - WT not defined by trigger memory - activated at cycle time $FF00_H$

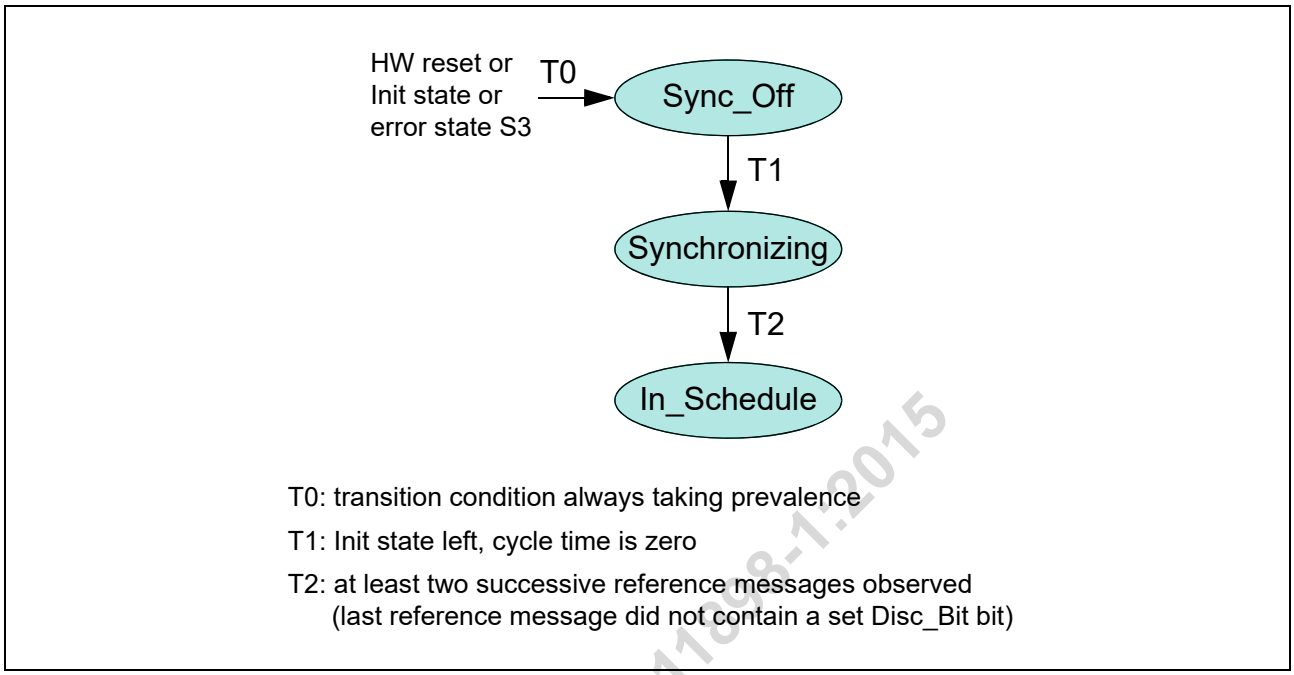
Level 0 operation is signalled via:

- MTTCAN0TTOST excluding bits AWE, WFE, GSI, GFI, RTO (no function)

ISO CANFD ISO 11898-1:2015

### (1) Synchronizing

**Figure 19.29** below describes the states and state transitions in TTCAN Level 0 operation. Level 0 has no In_Gap state.




ISO CANFD ISO 11898-1:2015

**Figure 19.29 Level 0 schedule synchronization state machine**

### (2) Handling of Error Levels

During Level 0 operation only the following error conditions may occur:

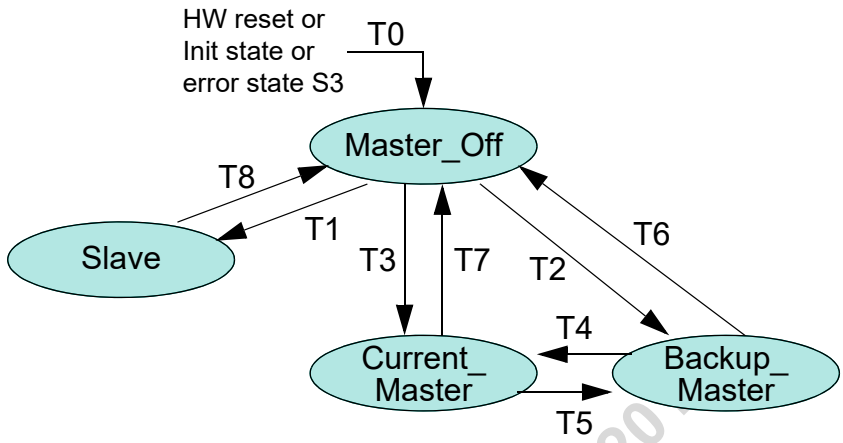
- Watch_Trigger_Reached (S3), reached cycle time $FF00_H$
- CAN_Bus_Off (S3)

Since no S1 and S2 error are possible, the error level can only switch between S0 (No Error) and S3 (Severe Error). In TTCAN Level 0 an S3 error is handled differently. When error level S3 is reached, both MTTCAN0TTOST.SYS and MTTCAN0TTOST.MS are reset, and interrupt flags MTTCAN0TTIR.GTE and MTTCAN0TTIR.GTD are set.

When error level S3 (MTTCAN0TTOST.EL = “11”) is entered, bus monitoring mode is, contrary to TTCAN Level 1 and Level 2, not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

### (3) Master Slave Relation

**Figure 19.30** below describes the master slave relation in TTCAN Level 0. In case of an S3 error the M_TTCAN returns to state Master_Off.




T0: transition condition always taking prevalence

T1: reference message observed when not potential time master

T2: reference message observed with master priority ≠ own master priority, error state ≠ S3

T3: reference message observed with master priority = own master priority, error state ≠ S3

T4: reference message observed with own master priority

T5: reference message observed with master priority higher than own master priority

T6: error state S3

T7: error state S3

T8: error state S3

ISO CANFD ISO 11898-1:20

**Figure 19.30 Level 0 master to slave relation**

## 19.6.4.10 Asynchronous Serial Communication

When configured for TTCAN Level 1 or Level 2 operation, the M_TTCAN time base can be used to switch access to the CAN bus for predefined time windows between M_TTCAN and an ASC module (see **Figure 19.31**).

When an exclusive time window is assigned to the ASC module, the multiplexer connects the ASC module to the CAN transceiver. ASC transmission is free of CAN requirements for arbitration and fault confinement and therefore higher bit rates and effective payloads are possible

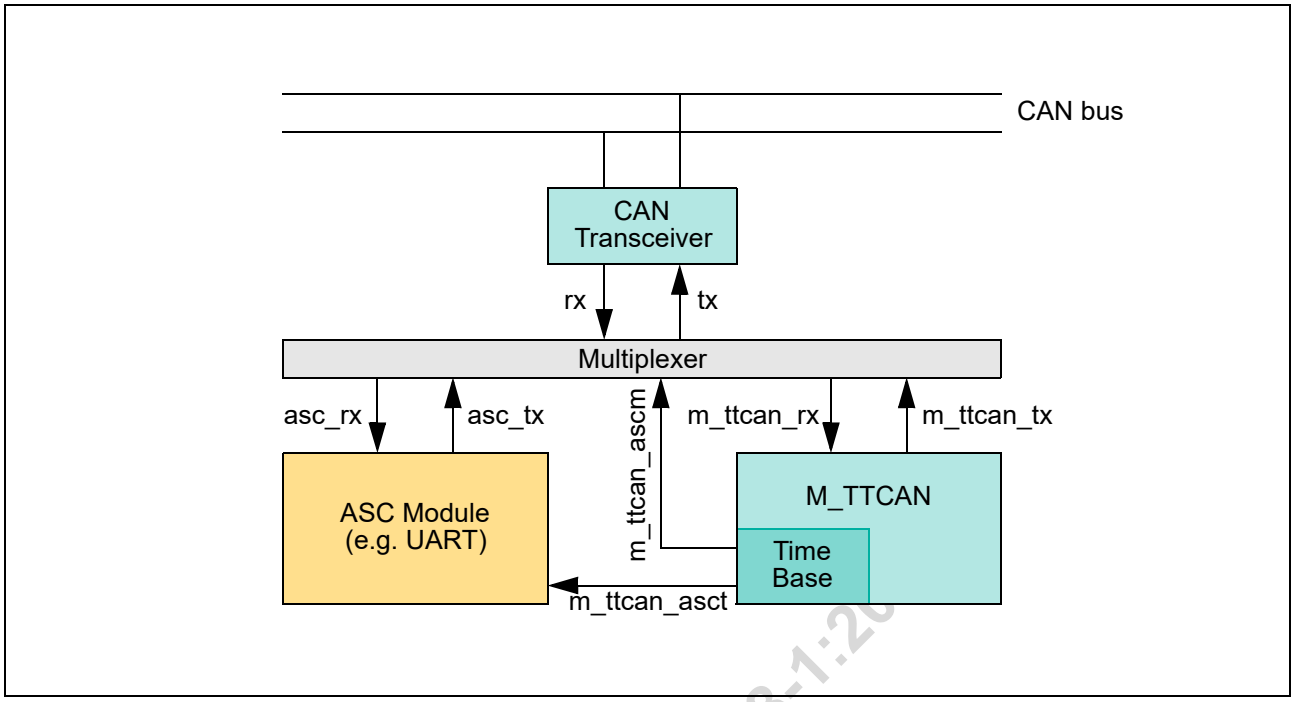



**Figure 19.31 Asynchronous Serial Communication at CAN**

Asynchronous serial operation is configured for each trigger element separately via T0.ASC. The M_TTCAN's time base controls access to the CAN transceiver for M_TTCAN or ASC module via output signal m_ttcan_ascm. Output signal m_ttcan_asct controls whether the ASC module is transmitter or receiver. For ASC transmission only one node in the network must be configured as transmitter while all other nodes are receivers.

With T0.ASC = "00" the ASC module is disconnected from the CAN bus (m_ttcan_ascm = '0', m_ttcan_asct = '0'). When T0.ASC = "01" the ASC module is receiver (m_ttcan_ascm = '1', m_ttcan_asct = '0'). When T0.ASC = "10" the ASC module is transmitter (m_ttcan_ascm = '1', m_ttcan_asct = '1')

ISO CANFD ISO 11898-1:20

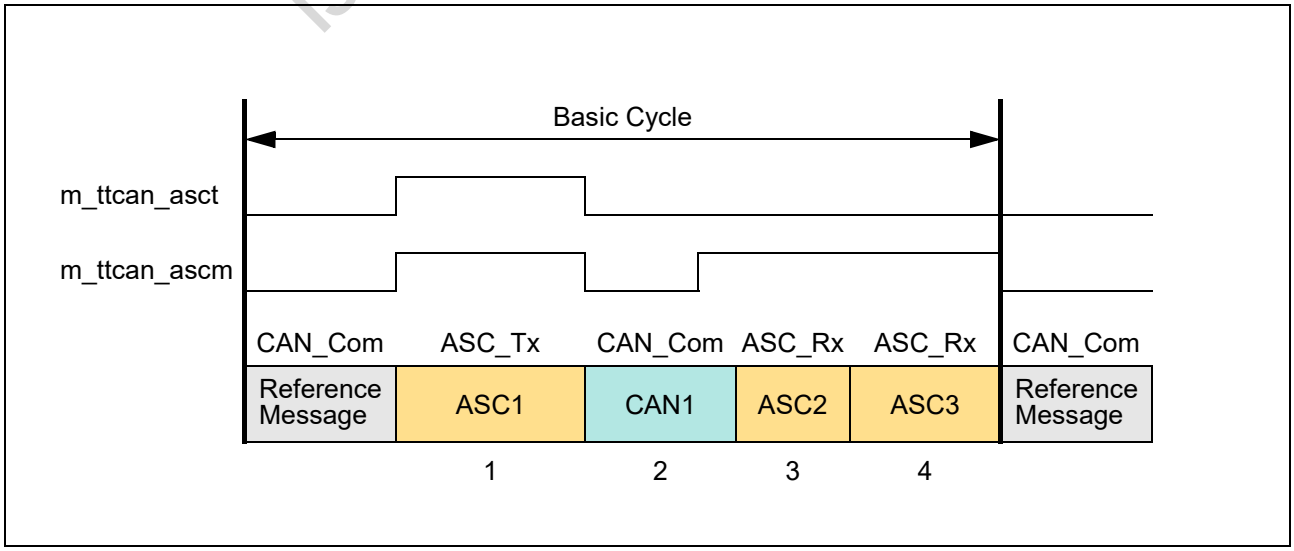



**Figure 19.32 ASC@CAN operation**

There are separate time windows for ASC transmission and reception. For each ASC time window there is one predefined transmitter, all other nodes are configured as receivers.

To start with an ASC window, the Host has to set CCCR.ASM during initialization. The bit is reset at the end of the first ASC window after the M_TTCAN has finished initializing. During time-triggered operation CCCR.ASM is set by the M_TTCAN at the beginning of an ASC window (trigger memory element with T0.ASC = “10”, “11”). It is reset by each trigger memory element with T0.ASC = “00”.

When CCCR.ASM is set, the CAN protocol controller

- sends no error/overload frames
- does not increment ECR.REC and ECR.TEC
- waits for Bus_Idle (11 recessive bits) after it has detected an error or overload condition
- acknowledges valid frames
- may start the transmission of a frame after it has detected Bus_Idle (11 recessive bits)

ASC mode operation is only entered when the M_TTCAN is synchronization state In_Schedule or In_Gap (m_ttcan_ascm = ’0’ and m_ttcan_asct = ’0’ when not synchronized).

### 19.6.4.11 Synchronization to external Time Schedule

This feature can be used to synchronize the phase of the M_TTCAN’s schedule to an external schedule (e.g. that of a second TTCAN network or FlexRay network). It is applicable only when the M_TTCAN is current time master (MTTCAN0TTOST.MS = “11”).

External synchronization is controlled by event trigger input pin m_ttcan_evt (MTTCAN0EVT) . If bit MTTCAN0TTOCN.ESCN is set, a rising edge at pin m_ttcan_evt (MTTCAN0EVT) the M_TTCAN compares its actual cycle time with the target phase value configured by MTTCAN0TTGTP.CTP.

Before setting MTTCAN0TTOCN.ESCN the Host has to adapt the phases of the two time schedules e.g. by using the TTCAN gap control (see **Section 19.6.4.3, TTCAN Gap Control**). When the Host sets MTTCAN0TTOCN.ESCN, MTTCAN0TTOST.SPL is set.

If the difference between the cycle time and the target phase value MTTCAN0TTGTP.CTP at the rising edge at pin m_ttcan_evt (MTTCAN0EVT) is greater than 9 NTU, the phase lock bit MTTCAN0TTOST.SPL is reset, and interrupt flag MTTCAN0TTIR.CSM is set. MTTCAN0TTOST.SPL is also reset (and MTTCAN0TTIR.CSM is set), when another node becomes time master.

If both MTTCAN0TTOST.SPL and MTTCAN0TTOCN.ESCN are set, and if the difference between the cycle time and the target phase value MTTCAN0TTGTP.CTP at the rising edge at pin m_ttcan_evt (MTTCAN0EVT) is less or equal 9 NTU, the phase lock bit MTTCAN0TTOST.SPL remains set, and the measured difference is used as reference trigger offset value to adjust the phase at the next transmitted reference message.

**NOTE**

The rising edge detection at pin m_ttcan_evt (MTTCAN0EVT) is enabled with the start of each basic cycle. The first rising edge triggers the compare of the actual cycle time with MTTCAN0TTGTP.CTP. All further edges until the beginning of the next basic cycle are ignored.

ISO CANFD ISO 11898-1:2015

## 19.7 Usage Notes

### 19.7.1 Port sharing between M_CAN IP and RLIN

By port sharing, ASC@CAN functionality can be achieved by sharing the M_TTCAN/M_CAN with the RLIN3.

All MTTCAN channel 0 pins will be shared with RLIN30.

One pair of MCAN channel 0 pins will be shared with RLIN30.

See the **Section 2, Pin Functions** and the **Section 18, LIN/UART Interface (RLIN3)** for details.

## 19.8 CAUTION

| No. | Outline |
|---|---|
| ID1 | M(TT)CAN - Edge filtering causes mis-synchronization when falling edge at Rx input pin coincides with end of integration phase.<br>(Direction of use) |
| ID2 | M(TT)CAN - Retransmission in DAR mode due to lost arbitration at the first two identifier bits.<br>(Direction of use) |

ISO CANFD ISO 11898-1:2015

| | | |
|---|---|---|
| ID1 | | M(TT)CAN - Edge filtering causes mis-synchronization when falling edge at Rx input pin coincides with end of integration phase. |
| | Description | When edge filtering is enabled (M(TT)CANnCCCR.EFBI = '1') and when the end of the integration phase coincides with a falling edge at the Rx input pin it may happen, that the M(TT)CAN synchronizes itself wrongly and does not correctly receive the first bit of the frame. In this case the CRC will detect that the first bit was received incorrectly, it will rate the received FD frame as faulty and an error frame will be send.<br><br>The issue only occurs, when there is a falling edge at the Rx input pin (m(tt)can_rx) within the last time quantum (tq) before the end of the integration phase. The last time quantum of the integration phase is at the sample point of the 11th recessive bit of the integration phase. When the edge filtering is enabled, the bit timing logic of the M(TT)CAN sees the Rx input signal delayed by the edge filtering. When the integration phase ends, the edge filtering is automatically disabled. This affects the reset of the FD CRC registers at the beginning of the frame. The Classical CRC register are not affected, so this issue does not affect the reception of Classical frames.<br><br>In CAN communication, the M(TT)CAN may enter integrating state (either by resetting M(TT)CANnCCCR.INIT or by protocol exception event) while a frame is active on the bus. In this case the 11 recessive bits are counted between the acknowledge bit and the following start of frame. All nodes have synchronized at the beginning of the dominant acknowledge bit. This means that the edge of the following start of frame bit cannot fall on the sample point, so the issue does not occur. The issue occurs only when the M(TT)CAN is, by local errors, mis-synchronized with regard to the other nodes.<br><br>Glitch filtering as specified in ISO 11898-1:2015 is fully functional.<br><br>Edge filtering was introduced for applications where the data bit time is at least two tq (of the nominal bit time) long. In that case, edge filtering requires at least two consecutive dominant time quanta before the counter counting the 11 recessive bits for idle detection is restarted. This means edge filtering covers the theoretical case of occasional 1-tq-long dominant spikes on the CAN bus that would delay idle detection. Repeated dominant spikes on the CAN bus would disturb all CAN communication, so the filtering to speed up<br>idle detection would not help network performance.<br><br>When this rare event occurs, the M(TT)CAN sends an error frame and the sender of the affected frame retransmits the frame. When the retransmitted frame is received, the M(TT)CAN has left integration phase and the frame will be received correctly. Edge filtering is only applied during integration phase, it is never used during normal operation. As integration phase is very short with respect to "active communication time", the impact on total error frame rate is negligible. The issue has no impact on data integrity.<br><br>The M(TT)CAN enters integration phase under the following conditions:<br>• when M(TT)CANnCCCR.INIT is set to '0' after start-up<br>• after a protocol exception event (only when M(TT)CANnCCCR.PXHD = '0')<br><br>Scope:<br>The erratum is limited to FD frame reception when edge filtering is active (M(TT)CANnCCCR.EFBI = '1') and when the end of the integration phase coincides with a falling edgeat the Rx input pin.<br><br>Effect:<br>The calculated CRC value does not match the CRC value of the received FD frame and the M(TT)CAN sends an error frame. After retransmission the frame is received correctly. |
| | Workaround | Disable edge filtering or wait on retransmission in case this rare event happens. |

ISO CAN FD ISO 11898-1:2015

| | | |
|---|---|---|
| ID2 | | M(TT)CAN - Retransmission in DAR mode due to lost arbitration at the first two identifier bits. |
| | Description | When the M(TT)CAN is configured in DAR mode (M(TT)CANnCCCR.DAR = '1') the Automatic Retransmission for transmitted messages that have been disturbed by an error or have lost arbitration is disabled. When the transmission attempt is not successful, the Tx Buffer's transmission request bit (M(TT)CANnTXBRP.TRPxx) shall be cleared and its cancellation finished bit (M(TT)CANnTXBCF.CFxx) shall be set.<br><br>When the transmitted message looses arbitration at one of the first two identifier bits, it may happen, that instead of the bits of the actually transmitted Tx Buffer, the M(TT)CANnTXBRP.TRPxx and M(TT)CANnTXBCF.CFxx bits of the previously started Tx Buffer (or Tx Buffer 0 if there is no previous transmission attempt) are written (M(TT)CANnTXBRP.TRPxx = '0', M(TT)CANnTXBCF.CFxx = '1').<br><br>If in this case the M(TT)CANnTXBRP.TRPxx bit of the Tx Buffer that lost arbitration at the first two identifier bits has not been cleared, retransmission is attempted.<br>When the M(TT)CAN looses arbitration again at the immediately following retransmission, then actually and previously transmitted Tx Buffer are the same and this Tx Buffer's M(TT)CANnTXBRP.TRPxx bit is cleared and its M(TT)CANnTXBCF.CFxx bit is set.<br><br>Scope:<br>The erratum is limited to the case when the M(TT)CAN looses arbitration at one of the first two transmitted identifier bits while in DAR mode.<br>The problem does not occur when the transmitted message has been disturbed by an error.<br><br>Effects:<br>In this case it may happen, that the M(TT)CANnTXBRP.TRPxx bit is cleared after the second transmision attempt instead of the first.<br>Additionally it may happen that the M(TT)CANnTXBRP.TRPxx bit of the previously started Tx Buffer is cleared, if it has been set again. As in this case the previously started Tx Buffer has lost M(TT)CAN internal arbitration against the active Tx Buffer, its message has a lower identifier priority. It would also have lost arbitration on the CAN bus at the same position. |
| | Workaround | None |

ISO CANFD ISO 11898-1:2015

## 19.9 Difference among P1M-C, P1H-C and P1H-CE

There are differences in channels, interrupts, and external pins. For details, please refer to **Table 19.2**, **Table 19.8**, and **Table 19.9**.

ISO CANFD ISO 11898-1:2015

# Section 20 FlexRay (FLX)

The FlexRay IP-module performs communication according to the FlexRay protocol specification v2.1. With maximum specified sample clock the bitrate is 10 MBit/s. Additional bus driver (BD) hardware is required for connection to the physical layer.

This section contains a generic description of the FlexRay.

The first part of this section describes all RH850/P1x-C specific properties, such as the number of channels, register base addresses, internal input/output signals, etc.

The remainder of this section describes the properties common to all versions of the RH850/P1x-C FlexRay.

**NOTE**

If not other mentioned the meaning of POC in **Section 20** is Protocol Operation Control.

## 20.1 Overview of RH850/P1x-C FlexRay

### 20.1.1 Channels

This microcontroller has the following number of FlexRay modules.

**Table 20.1 Number of FlexRay Modules**

| FlexRay | P1M-C, P1H-C (4MB, BGA-156) | P1H-C (4MB, BGA-292) and P1H-CE |
|---|---|---|
| Number of modules | 1 | 2 |
| Name | FLXn (n = 0) | FLXn (n = 0, 1) |

#### Index n

Throughout this section, the individual modules of the FlexRay is identified by the index “n” , for example, FLXAnFROC for the FlexRay control register.

### 20.1.2 Register Base Addresses

All FlexRay register addresses are given as address offsets to the individual base address<FLXn_base>.

The base address <FLXn_base> of each FlexRay is listed in the following table:

**Table 20.2 Register Base Addresses <FLXn_base>**

| FlexRay Channel | <FLXAn_base> Address |
|---|---|
| FLXA0 | $1002\ 0000_H$ |
| FLXA1 | $1002\ 1000_H$ |

### 20.1.3 Clock Supply

FlexRay provide one clock input.

**Table 20.3 Clock Supply**

| Module | Clock Name | Clock Symbol | Description |
|---|---|---|---|
| FLXAn | High Speed Peripheral clock 1 | CLKP_H1 | FlexRay sample clock |
| | Speed Peripheral clock 1 | CLKP_H1 | NTU clock Source |
| | High speed system clock | CLK_HSB | Bus clock |

Note 1. CLK_HSB must be set to min.44.8125MHz
Note 2. CLKP_H1 must be 80MHz

### 20.1.4 Interrupts

FlexRay can generate the following interrupt requests:

**Table 20.4 Interrupt Requests**

| Unit Interrupt Name | Outline | Interrupt Number | DMA Trigger Number | DTS Trigger Number |
|---|---|---|---|---|
| **FlexRay0** | | | | |
| INTFLX0LINE0 | Universal interrupt chA for FLX0 | 186 | — | — |
| INTFLX0LINE1 | Universal interrupt chB for FLX0 | 187 | — | — |
| INTFLX0TIM0 | Timer 0 interrupt for FLX0 | 188 | — | — |
| INTFLX0TIM1 | Timer 1 interrupt for FLX0 | 189 | — | — |
| INTFLX0TIM2 | Timer 2 interrupt for FLX0 | 190 | — | — |
| INTFLX0FDA | FIFO data available (FIFO not empty) interrupt for FLX0 | 191 | — | — |
| INTFLX0FW | FIFO warning interrupt for FLX0 | 192 | — | — |
| INTFLX0OW | Output transfer warning interrupt for FLX0 | 193 | — | — |
| INTFLX0OT | Output transfer done interrupt for FLX0 | 194 | — | — |
| INTFLX0IQF | Input queue full interrupt for FLX0 | 195 | — | — |
| INTFLX0IQE | Input queue empty interrupt for FLX0 | 196 | — | — |
| **FlexRay1** | | | | |
| INTFLX1LINE0 | Universal interrupt chA for FLX1 | 197 | — | — |
| INTFLX1LINE1 | Universal interrupt chB for FLX1 | 198 | — | — |
| INTFLX1TIM0 | Timer 0 interrupt for FLX1 | 199 | — | — |
| INTFLX1TIM1 | Timer 1 interrupt for FLX1 | 200 | — | — |
| INTFLX1TIM2 | Timer 2 interrupt for FLX1 | 201 | — | — |
| INTFLX1FDA | FIFO data available (FIFO not empty) interrupt for FLX1 | 202 | — | — |
| INTFLX1FW | FIFO warning interrupt for FLX1 | 203 | — | — |
| INTFLX1OW | Output transfer warning interrupt for FLX1 | 204 | — | — |
| INTFLX1OT | Output transfer done interrupt for FLX1 | 205 | — | — |
| INTFLX1IQF | Input queue full interrupt for FLX1 | 206 | — | — |
| INTFLX1IQE | Input queue empty interrupt for FLX1 | 207 | — | — |
| **NTU** | | | | |
| INTFLXNTU | NTU loss interrupt | 40 | — | — |

### 20.1.5 FlexRay Hardware Reset

FlexRay and registers of FlexRay are initialized by the following reset condition:

**Table 20.5 Reset Condition**

| Reset Name | Reset condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| All Registers | √ | √ | √ | √ | √ |

### 20.1.6 I/O Signals

The I/O signals of the FlexRay are listed in the following table.

**Table 20.6 FlexRay I/O Signals**

| Pin name | I/O | Function |
|---|---|---|
| FLXnRXDA | Input | Flex Ray n channel A receive data input |
| FLXnTXDA | Output | Flex Ray n channel A transmit data output |
| FLXnTXENA | Output | Flex Ray n channel A transmit data enable |
| FLXnRXDB | Input | Flex Ray n channel B receive data input |
| FLXnTXDB | Output | Flex Ray n channel B transmit data output |
| FLXnTXENB | Output | Flex Ray n channel B transmit data enable |
| FLXnSTPWT | Input | Flexray n stop watch trigger input |
| FLXNTUOUT | Output | NTU clock output |
| BHPDGRCLK1 | Output | Degraded clock |

### 20.1.7 Functions

For communication on a FlexRay network, individual message buffers with up to 254 data bytes are configurable. The message storage is a Message RAM that is configurable to hold up to 128 message buffers. All functions concerning the handling of messages are implemented in the Message Handler. Those functions are the acceptance filtering, the transfer of messages between the two FlexRay Channel Protocol Controllers and the Message RAM, maintaining the transmission schedule as well as providing message status information.

The register set of the FlexRay IP-module can be accessed directly by an external Host via the module's Host interface. These registers are used to control/configure/monitor the FlexRay Channel Protocol Controllers, Message Handler, Global Time Unit, System Universal Control, Frame and Symbol Processing, Network Management, Interrupt Control, to access the Message RAM via Input / Output Buffer; and to control the data transfer between the Message RAM and the Local RAM/Global RAM.

The FlexRay IP-module supports the following features:

| Item | Specification |
|---|---|
| Communication | Conformance with FlexRay protocol specification v2.1 |
| Data transfer rate | Up to 10 Mbit/s on each channel |
| Data link layer clock frequency | 80 MHz |
| FlexRay channels | 2 (channels A and B) |
| Message buffer | Up to 128 message buffers configurable |
| | Configuration of message buffers with different payload lengths possible |
| | Each message buffer can be configured as receive buffer, as transmit buffer or as part of the receive FIFO |
| | Filtering for slot counter, cycle counter, and channel |
| Message RAM | 8 Kbyte of Message RAM for storage of e.g. |
| | 128 message buffers with max. 48 byte data section |
| | Up to 30 message buffers with 254 byte data section |
| FIFO | One configurable receive FIFO |
| Message buffer access | By Host CPU via Input and Output Buffer<br>Input Buffer: Holds message to be transferred to the Message RAM<br>Output Buffer: Holds message read from the Message RAM |
| | By data transfer function<br>Input transfer: Message buffer content is transferred from Local RAM/Global RAM to Message RAM on CPU request<br>Output transfer: Message buffer content is transferred from Message RAM to Local RAM/Global RAM automatically |
| Network management | Automatic hardware support |
| Interrupts | Maskable module interrupts |
| Timer | 2 absolute timer<br>1 relative timer |
| NTU generation | Generation of FlexRay macrotick<br>Generation of FlexRay cycle |

## 20.1.8 Block Diagram

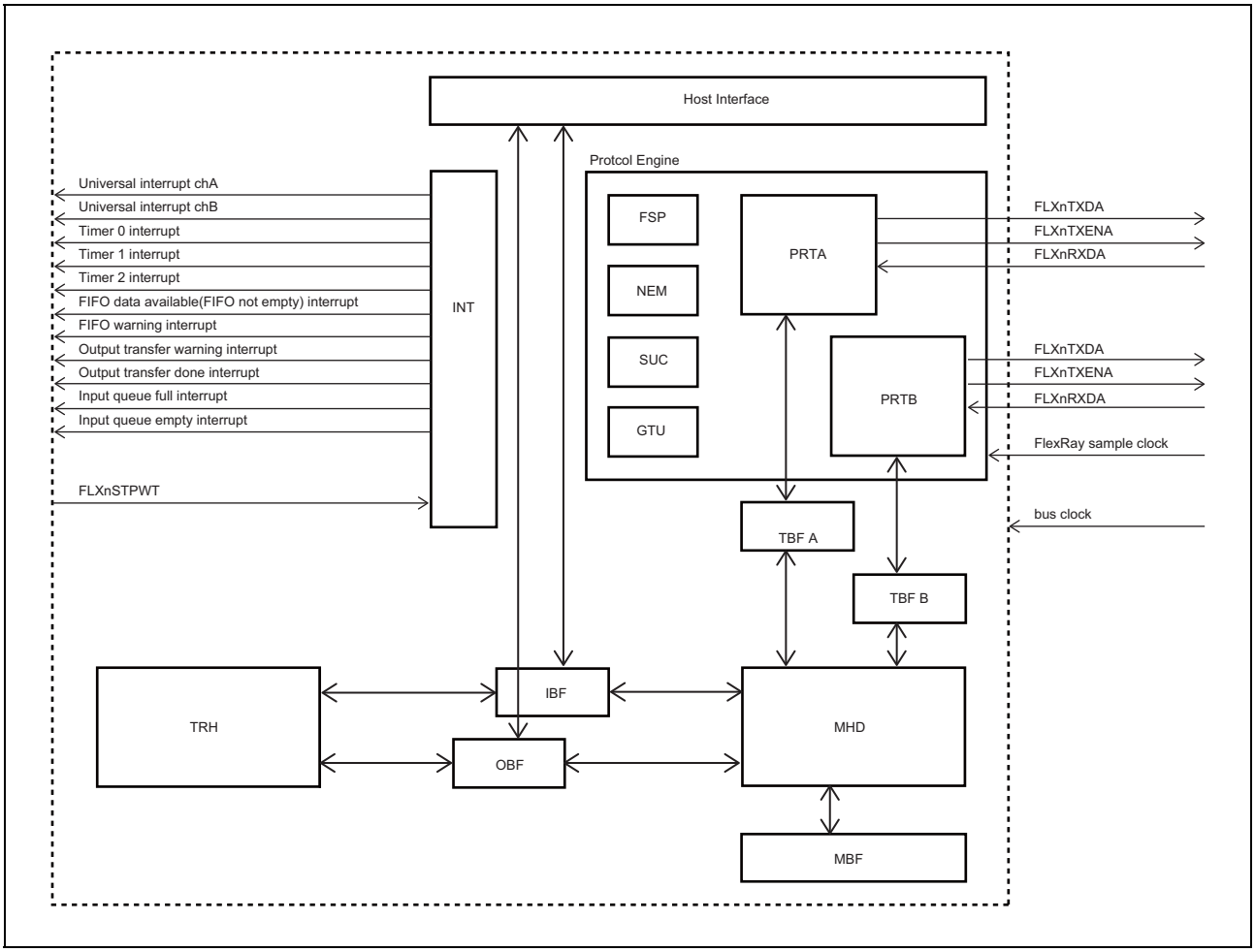



**Figure 20.1 FlexRay IP Block Diagram**

### Input Buffer (IBF)

For write access to the message buffers configured in the Message RAM, the Host can write the header and data section for a specific message buffer to the Input Buffer. The Message Handler then transfers the data from the Input Buffer to the selected message buffer in the Message RAM.

### Output Buffer (OBF)

For read access to a message buffer configured in the Message RAM the Message Handler transfers the selected message buffer to the Output Buffer. After the transfer has completed, the Host can read the header and data section of the transferred message buffer from the Output Buffer.

### Message Handler (MHD)

The FlexRay Message Handler controls data transfers between the following components:

- Input / Output Buffer and Message RAM
- Temporary buffer RAMs of the two FlexRay Protocol Controllers and Message RAM

### Message RAM (MRAM)

The Message RAM consists of a single-ported RAM that store up to 128 FlexRay message buffers together with the related configuration data (header and data partition).

### Temporary buffer RAM (TBF A/B)

Stores the data section of two complete messages.

### FlexRay Channel Protocol Controller (PRT A/B)

The FlexRay Channel Protocol Controllers consist of shift register and FlexRay protocol FSM. They are connected to the Temporary buffer RAMs for intermediate message storage and to the physical layer via bus driver BD.

They perform the following functionality:

- Control and check of bit timing
- Reception / transmission of FlexRay frames and symbols
- Check of header CRC
- Generation / check of frame CRC
- Interfacing to bus driver

### Global Time Unit (GTU)

The Global Time Unit performs the following functions:

- Generation of microtick (µT)
- Generation of macrotick (MT)
- Fault tolerant clock synchronization by FTM algorithm
  - rate correction
  - offset correction
- Cycle counter
- Timing control of static segment
- Timing control of dynamic segment (minislotting)
- Support of external clock correction

### System Universal Control (SUC)

The System Universal Control controls the following functions:

- Configuration
- Wakeup
- Startup
- Normal Operation
- Passive Operation

### Frame and Symbol Processing (FSP)

The Frame and Symbol Processing controls the following functions:

- Checks the correct timing of frames and symbols
- Tests the syntactical and semantical correctness of received frames
- Sets the slot status flags

### Network Management (NEM)

Handles the network management vector.

### Interrupt Control (INT)

The Interrupt Controller performs the following functions:

- Provides error and status interrupt flags
- Enable / disable interrupt sources
- Assignment of interrupt sources to one of the two general module interrupt lines
- Enable / disable module interrupt lines

### Timer (TIM)

The Timer module includes the following macrotick timer:

- two absolute timers
- one relative timer
- one stop watch timer

### Transfer Handler (TRH)

Handles the data transfer between Local RAM/Global RAM and FlexRay module.

The Transfer Handler supports the following transfer types:

- Transfer of buffer configuration data from the Local RAM/Global RAM to the Message RAM
- Transfer of payload data for temporary buffers from the Local RAM/Global RAM to the Message RAM
- Transfer of buffer configuration data and payload data for temporary buffer from the Local RAM/Global RAM to the Message RAM
- Automatic transfer of payload data from receive buffer to the Local RAM/Global RAM upon frame reception
- Automatic transfer of payload data, buffer configuration data and message buffer status data from receive buffer to the Local RAM/Global RAM upon frame reception
- Automatic transfer of buffer configuration data and message buffer status data from the dedicated transmit/receive buffer to the Local RAM/Global RAM in response to slot status update
- Manual transfer of payload data, buffer configuration data and message buffer status data from the dedicated transmit/receive buffer to the Local RAM/Global RAM

# 20.2 Register

## 20.2.1 Register Map

The FlexRay module allocates an address space as shown in **Table 20.7, FlexRay Register Map**.

Within this specification the “Values after reset” refers to the microcontroller’s HW reset. For registers in the address range $0010_H$ to $0FFF_H$ the “Value after reset” are also applicable when the SW reset (using FLXAnFROC.OE) is applied.

The addresses in this specification are listed as offsets from a base address. The base address <FLXn_base> must thus be added to the addresses.

For <FLXn_base>, see **Section 20.1.2, Register Base Addresses**.

**Table 20.7 FlexRay Register Map (1/5)**

| Register Name | Symbol | Value after Reset | Address | Access Size | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | HBG/PBG | Other |
| FlexRay Operation Control Register | FLXAnFRPV | 2A10 $2300_H$ | <FLXn_base> + $0000_H$ | 32 | HBG. FlexRayn*1 | — |
| FlexRay Operation Control Register | FLXAnFROC | 0000 $0000_H$ | <FLXn_base> + $0004_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Operation Status Register | FLXAnFROS | 0000 $0000_H$ | <FLXn_base> + $000C_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Lock Register | FLXAnFRLCK | 0000 $0000_H$ | <FLXn_base> + $001C_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Error Interrupt Register | FLXAnFREIR | 0000 $0000_H$ | <FLXn_base> + $0020_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Status Interrupt Register | FLXAnFRSIR | 0000 $0000_H$ | <FLXn_base> + $0024_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Error Interrupt Line Select | FLXAnFREILS | 0000 $0000_H$ | <FLXn_base> + $0028_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Status Interrupt Line Select | FLXAnFRSILS | 0303 $FFFF_H$ | <FLXn_base> + $002C_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Error Interrupt Enable Set Register | FLXAnFREIES | 0000 $0000_H$ | <FLXn_base> + $0030_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Error Interrupt Enable Reset Register | FLXAnFREIER | 0000 $0000_H$ | <FLXn_base> + $0034_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Status Interrupt Enable Set Register | FLXAnFRSIES | 0000 $0000_H$ | <FLXn_base> + $0038_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Status Interrupt Enable Reset Register | FLXAnFRSIER | 0000 $0000_H$ | <FLXn_base> + $003C_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Interrupt Line Enable Register | FLXAnFRILE | 0000 $0000_H$ | <FLXn_base> + $0040_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Timer 0 Configuration Register | FLXAnFRT0C | 0000 $0000_H$ | <FLXn_base> + $0044_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Timer 1 Configuration Register | FLXAnFRT1C | 0002 $0000_H$ | <FLXn_base> + $0048_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Stop Watch Register 1 | FLXAnFRSTPW1 | 0000 $0000_H$ | <FLXn_base> + $004C_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Stop Watch Register 2 | FLXAnFRSTPW2 | 0000 $0000_H$ | <FLXn_base> + $0050_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay SUC Configuration Register 1 | FLXAnFRSUCC1 | 0C40 $1080_H$ | <FLXn_base> + $0080_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay SUC Configuration Register 2 | FLXAnFRSUCC2 | 0100 $0504_H$ | <FLXn_base> + $0084_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |

Table 20.7 FlexRay Register Map (2/5)

| Register Name | Symbol | Value after Reset | Address | Access Size | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | HBG/PBG | Other |
| FlexRay SUC Configuration Register 3 | FLXAnFRSUCC3 | 0000 0011$_H$ | <FLXn_base> + 0088$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay NEM Configuration Register | FLXAnFRNEMC | 0000 0000$_H$ | <FLXn_base> + 008C$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay PRT Configuration Register 1 | FLXAnFRPRTC1 | 084C 0633$_H$ | <FLXn_base> + 0090$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay PRT Configuration Register 2 | FLXAnFRPRTC2 | 0F2D 0A0E$_H$ | <FLXn_base> + 0094$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay MHD Configuration Register | FLXAnFRMHDC | 0000 0000$_H$ | <FLXn_base> + 0098$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay GTU Configuration Register 1 | FLXAnFRGTUC1 | 0000 0280$_H$ | <FLXn_base> + 00A0$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay GTU Configuration Register 2 | FLXAnFRGTUC2 | 0002 000A$_H$ | <FLXn_base> + 00A4$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay GTU Configuration Register 3 | FLXAnFRGTUC3 | 0202 0000$_H$ | <FLXn_base> + 00A8$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay GTU Configuration Register 4 | FLXAnFRGTUC4 | 0008 0007$_H$ | <FLXn_base> + 00AC$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay GTU Configuration Register 5 | FLXAnFRGTUC5 | 0E00 0000$_H$ | <FLXn_base> + 00B0$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay GTU Configuration Register 6 | FLXAnFRGTUC6 | 0002 0000$_H$ | <FLXn_base> + 00B4$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay GTU Configuration Register 7 | FLXAnFRGTUC7 | 0002 0004$_H$ | <FLXn_base> + 00B8$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay GTU Configuration Register 8 | FLXAnFRGTUC8 | 0000 0002$_H$ | <FLXn_base> + 00BC$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay GTU Configuration Register 9 | FLXAnFRGTUC9 | 0000 0101$_H$ | <FLXn_base> + 00C0$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay GTU Configuration Register 10 | FLXAnFRGTUC10 | 0002 0005$_H$ | <FLXn_base> + 00C4$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay GTU Configuration Register 11 | FLXAnFRGTUC11 | 0000 0000$_H$ | <FLXn_base> + 00C8$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay CC Status Vector Register | FLXAnFRCCSV | 0010 4000$_H$ | <FLXn_base> + 0100$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay CC Error Vector Register | FLXAnFRCCEV | 0000 0000$_H$ | <FLXn_base> + 0104$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay Slot Counter Value Register | FLXAnFRSCV | 0000 0000$_H$ | <FLXn_base> + 0110$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay Macrotick and Cycle Counter Value Register | FLXAnFRMTCCV | 0000 0000$_H$ | <FLXn_base> + 0114$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay Rate Correction Value Register | FLXAnFRRCV | 0000 0000$_H$ | <FLXn_base> + 0118$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay Offset Correction Value Register | FLXAnFROCV | 0000 0000$_H$ | <FLXn_base> + 011C$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay Sync Frame Status Register | FLXAnFRSFS | 0000 0000$_H$ | <FLXn_base> + 0120$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay Symbol Window and NIT Status Register | FLXAnFRSWNIT | 0000 0000$_H$ | <FLXn_base> + 0124$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |
| FlexRay Aggregated Channel Status Register | FLXAnFRACS | 0000 0000$_H$ | <FLXn_base> + 0128$_H$ | 8, 16, 32 | HBG. FlexRayn*[1] | — |

Table 20.7 FlexRay Register Map (3/5)

| Register Name | Symbol | Value after Reset | Address | Access Size | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | HBG/PBG | Other |
| FlexRay Even Sync ID Register m (m = 1 to 15) | FLXAnFRESIDm (m = 1 to 15) | 0000 $0000_H$ | <FLXn_base> + $0130_H$ to <FLXn_base> + $0168_H$ (<FLXn_base> + $0130_H$ + (m – 1)*4) | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Odd Sync ID Register m (m = 1 to 15) | FLXAnFROSIDm (m = 1 to 15) | 0000 $0000_H$ | <FLXn_base> + $0170_H$ to <FLXn_base> + $01A8_H$ (<FLXn_base> + $0170_H$ + (m – 1)*4) | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Network Management Vector Register m (m = 1 to 3) | FLXAnFRNMVm (m = 1 to 3) | 0000 $0000_H$ | <FLXn_base> + $01B0_H$ to <FLXn_base> + $01B8_H$ (<FLXn_base> + $01B0_H$ + (m – 1)*4) | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Message RAM Configuration Register | FLXAnFRMRC | 0180 $0000_H$ | <FLXn_base> + $0300_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay FIFO Rejection Filter Register | FLXAnFRFRF | 0180 $0000_H$ | <FLXn_base> + $0304_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay FIFO Rejection Filter Mask Register | FLXAnFRFRFM | 0000 $0000_H$ | <FLXn_base> + $0308_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay FIFO Critical Level Register | FLXAnFRFCL | 0000 $0080_H$ | <FLXn_base> + $030C_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Message Handler Status Register | FLXAnFRMHDS | 0000 $0080_H$ | <FLXn_base> + $0310_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Last Dynamic Transmit Slot Register | FLXAnFRLDTS | 0000 $0000_H$ | <FLXn_base> + $0314_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay FIFO Status Register | FLXAnFRFSR | 0000 $0000_H$ | <FLXn_base> + $0318_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Message Handler Constraints Flags Register | FLXAnFRMHDF | 0000 $0000_H$ | <FLXn_base> + $031C_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Transmission Register i (i = 1 to 4) | FLXAnFRTXRQi (i = 1 to 4) | 0000 $0000_H$ | <FLXn_base> + $0320_H$ to <FLXn_base> + $032C_H$ (<FLXn_base> + $0320_H$ + (i – 1)*4) | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay New Data Register i (i = 1 to 4) | FLXAnFRNDATi (i = 1 to 4) | 0000 $0000_H$ | <FLXn_base> + $0330_H$ to <FLXn_base> + $033C_H$ (<FLXn_base> + $0330_H$ + (i – 1)*4) | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Message Buffer Status Changed Register i (i = 1 to 4) | FLXAnFRMBSCi (i = 1 to 4) | 0000 $0000_H$ | <FLXn_base> + $0340_H$ to <FLXn_base> + $034C_H$ (<FLXn_base> + $0340_H$ + (i – 1)*4) | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Write Data Section Register x (x = 1 to 64) | FLXAnFRWRDSx (x = 1 to 64) | 0000 $0000_H$ | <FLXn_base> + $0400_H$ to <FLXn_base> + $04FC_H$ (<FLXn_base> + $0400_H$ + (x – 1)*4) | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Write Header Section Register 1 | FLXAnFRWRHS1 | 0000 $0000_H$ | <FLXn_base> + $0500_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Write Header Section Register 2 | FLXAnFRWRHS2 | 0000 $0000_H$ | <FLXn_base> + $0504_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |

Table 20.7 FlexRay Register Map (4/5)

| Register Name | Symbol | Value after Reset | Address | Access Size | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | HBG/PBG | Other |
| FlexRay Write Header Section Register 3 | FLXAnFRWRHS3 | 0000 $0000_H$ | <FLXn_base> + $0508_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Input Buffer Command Mask Register | FLXAnFRIBCM | 0000 $0000_H$ | <FLXn_base> + $0510_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Input Buffer Command Request Register | FLXAnFRIBCR | 0000 $0000_H$ | <FLXn_base> + $0514_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Read Data Section Register x (x = 1 to 64) | FLXAnFRRDDSx (x = 1 to 64) | 0000 $0000_H$ | <FLXn_base> + $0600_H$ to <FLXn_base> + $06FC_H$ (<FLXn_base> + $0600_H$ + (x – 1)*4) | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Read Header Section Register 1 | FLXAnFRRDHS1 | 0000 $0000_H$ | <FLXn_base> + $0700_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Read Header Section Register 2 | FLXAnFRRDHS2 | 0000 $0000_H$ | <FLXn_base> + $0704_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Read Header Section Register 3 | FLXAnFRRDHS3 | 0000 $0000_H$ | <FLXn_base> + $0708_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Message Buffer Status Register | FLXAnFRMBS | 0000 $0000_H$ | <FLXn_base> + $070C_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Output Buffer Command Mask Register | FLXAnFROBCM | 0000 $0000_H$ | <FLXn_base> + $0710_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Output Buffer Command Request Register | FLXAnFROBCR | 0000 $0000_H$ | <FLXn_base> + $0714_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Input Transfer Configuration Register | FLXAnFRITC | 0000 $0000_H$ | <FLXn_base> + $0800_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Output Transfer Configuration Register | FLXAnFROTC | 0000 $0000_H$ | <FLXn_base> + $0804_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Input pointer table Base Address Register | FLXAnFRIBA | 0000 $0000_H$ | <FLXn_base> + $0808_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay FIFO pointer table Base Address Register | FLXAnFRFBA | 0000 $0000_H$ | <FLXn_base> + $080C_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Output pointer table Base Address Register | FLXAnFROBA | 0000 $0000_H$ | <FLXn_base> + $0810_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Input Queue Control Register | FLXAnFRIQC | 0000 $0000_H$ | <FLXn_base> + $0814_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay User Input transfer Request Register | FLXAnFRUIR | 0000 $0000_H$ | <FLXn_base> + $0818_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay User Output transfer Request Register | FLXAnFRUOR | 0000 $0000_H$ | <FLXn_base> + $081C_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Input Transfer Status Register | FLXAnFRITS | 0000 $0000_H$ | <FLXn_base> + $0820_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Output Transfer Status Register | FLXAnFROTS | 0000 $0000_H$ | <FLXn_base> + $0824_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Access Error Status Register | FLXAnFRAES | 0000 $0000_H$ | <FLXn_base> + $0828_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay Access Error Address Register | FLXAnFRAEA | 0000 $0000_H$ | <FLXn_base> + $082C_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| FlexRay message Data Available Register i (i = 0 to 3) | FLXAnFRDAi (i = 0 to 3) | 0000 $0000_H$ | <FLXn_base> + $0830_H$ to <FLXn_base> + $083C_H$ (<FLXn_base> + $0830_H$ + i * 4) | 8, 16, 32 | HBG. FlexRayn*1 | — |

Table 20.7 FlexRay Register Map (5/5)

| Register Name | Symbol | Value after Reset | Address | Access Size | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | HBG/PBG | Other |
| FlexRay Timer 2 Configuration Register | FLXAnFRT2C | 0000 $0000_H$ | <FLXn_base> + $0844_H$ | 8, 16, 32 | HBG. FlexRayn*1 | — |
| NTU prescaler control register | NTU0PRSCTL | $00_H$ | $FFF9C008_H$ | 8 | PBG3#0. PG3-FLXNTU | — |
| NTU prescaler compare register | NTU0PRSCMP | $00_H$ | $FFF9C00C_H$ | 8 | PBG3#0. PG3-FLXNTU | — |
| NTU backup clock prescaler setting register | NTU0BPRSCTL | $00_H$ | $FFF9C108_H$ | 8 | PBG3#0. PG3-FLXNTU | — |
| NTU backup clock prescaler compare register | NTU0BPRSCMP | $00_H$ | $FFF9C10C_H$ | 8 | PBG3#0. PG3-FLXNTU | — |
| NTU compare unit prescaler setting register | NTU0CPRSCTL | $00_H$ | $FFF9C208_H$ | 8 | PBG3#0. PG3-FLXNTU | — |
| NTU compare unit prescaler compare register | NTU0CPRSCMP | $00_H$ | $FFF9C20C_H$ | 8 | PBG3#0. PG3-FLXNTU | — |
| NTU compare unit control register | NTU0CCTL0 | $00_H$ | $FFF9C800_H$ | 8 | PBG3#0. PG3-FLXNTU | — |
| NTU compare unit threshold register L | NTU0CCMPL | $0001_H$ | $FFF9C808_H$ | 16 | PBG3#0. PG3-FLXNTU | — |
| NTU compare unit threshold register H | NTU0CCMPH | $03FF_H$ | $FFF9C80C_H$ | 16 | PBG3#0. PG3-FLXNTU | — |
| NTU compare unit protection command register | NTU0CPCMD | $00_H$ | $FFF9C810_H$ | 8 | PBG3#0. PG3-FLXNTU | — |
| NTU compare unit protection status register | NTU0CPS | $00_H$ | $FFF9C814_H$ | 8 | PBG3#0. PG3-FLXNTU | — |
| NTU loss monitor reset register | NTU0RST | 0000 $0000_H$ | $FFF9CC00_H$ | 32 | PBG3#0. PG3-FLXNTU | — |
| NTU configuration register | NTU0CFG | 0000 $0000_H$ | $FFF9CC04_H$ | 32 | PBG3#0. PG3-FLXNTU | — |
| NTU enable register | NTU0CLMEN | 0000 $0000_H$ | $FFF9CC08_H$ | 32 | PBG3#0. PG3-FLXNTU | — |
| NTU status register | NTU0MON | 0000 $0000_H$ | $FFF9CC0C_H$ | 32 | PBG3#0. PG3-FLXNTU | — |

Note 1. In the case of
n = 0 HBG.FlexRay0
n = 1 HBG.FlexRay1

## 20.2.2 FlexRay Operation register

### 20.2.2.1 FLXAnFROC — FlexRay Operation Control Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0004_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | T2IE | T1IE | T0IE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | OEP | — | — | — | — | — | BEC | OE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R/W | R/W |

**Table 20.8 FLXAnFROC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 18 | T2IE | Timer 2 interrupt enable Bit<br>0: Disabled<br>1: Enabled |
| 17 | T1IE | Timer 1 interrupt enable Bit<br>0: Disabled<br>1: Enabled |
| 16 | T0IE | Timer 0 interrupt enable Bit<br>0: Disabled<br>1: Enabled |
| 15 to 8 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 7 | OEP | Operation Enable bit Protection Bit<br>0: OE is unprotected<br>1: OE is protected |
| 6 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | BEC | Byte Endian Control Bit<br>0: Little endian<br>1: Big endian |
| 0 | OE | Operation Enable Bit<br>0: Operation disabled, SW reset<br>1: Operation Enabled |

**(1) FLXAnFROC.T2IE**

Timer 2 interrupt enable Bit

This bit controls the timer 2 interrupt.

0: Disabled

No interrupt will be requested and the timer 2 interrupt line will be released if pending.

1: Enabled

Timer 2 interrupt will be asserted when FLXAnFROTS.T2IS is 1.

**(2) FLXAnFROC.T1IE**

Timer 1 interrupt enable Bit

The user should only set this bit to 1 when timer 1 interrupt is not enabled in the FlexRay Status interrupt enable register (FLXAnFRSIES.TI1E should be 0).

This bit controls the timer 1 interrupt.

0: Disabled

No interrupt will be requested and the timer 1 interrupt line will be released if pending.

1: Enabled

Timer 1 interrupt will be asserted when FLXAnFROTS.T1IS is 1.

**(3) FLXAnFROC.T0IE**

Timer 0 interrupt enable Bit

The user should only set this bit to 1 when timer 0 interrupt is not enabled in the FlexRay Status interrupt enable register (FLXAnFRSIES.TI0E should be 0).

This bit controls the timer 0 interrupt.

0: Disabled

No interrupt will be requested and the timer 0 interrupt line will be released if pending.

1: Enabled

Timer 0 interrupt will be asserted when FLXAnFROTS.T0IS is 1.

**(4) FLXAnFROC.OEP**

Operation Enable bit Protection Bit

This bit protects against unintended write access to the OE bit.

0: OE is unprotected

Write access to the OE bit is enabled

1: OE is protected

Write access to the OE bit is disabled

### (5) FLXAnFROC.BEC

Byte Endian Control Bit

The user should only change this bit when FLXAnFROS.OS is 1.

This bit controls the byte order on reading and writing the FlexRay Network Management Vector register (FLXAnFRNMVn), FlexRay Write Data Section (FLXAnFRWRDSn) and FlexRay Read Data Section (FLXAnFRRDDSn). This bit also controls the byte order when reading or writing FlexRay payload data using the data transfer function.

For details about the byte alignment see **Section 20.3.17, Byte Alignment**.

0: Little endian

Byte alignment in FLXAnFRNMVn, FLXAnFRWRDSn and FLXAnFRRDDSn is in little endian style.

1: Big endian

Byte alignment in FLXAnFRNMVn, FLXAnFRWRDSn and FLXAnFRRDDSn is in big endian style.

### (6) FLXAnFROC.OE

Operation Enable Bit

The user can only write to this bit when FLXAnFROC.OEP is 0.

The user should only write this bit with 0 when FLXAnFROS.OS is 1.

The user should only write this bit with 1 when FLXAnFROS.OS is 0 and the FlexRay sample clock is enabled.

This bit controls the operation state and serves the software reset of the FlexRay module. The operation status bit (FLXAnFROS.OS) indicates whether the FlexRay module is in reset state or not.

0: Operation disabled, SW reset

Forcibly moves the FlexRay module to its reset state, whatever the state of the FlexRay module is.

1: Operation Enabled

Reset state of the FlexRay module is released.

### 20.2.2.2 FLXAnFROS — FlexRay Operation Status Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $000C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | T2IS | T1IS | T0IS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.9 FLXAnFROS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 18 | T2IS | Timer 2 Interrupt Status Bit<br>0: Timer 2 has not matched the conditions configured in the FLXAnFRT2C register<br>1: Timer 2 matched the conditions configured in the FLXAnFRT2C register |
| 17 | T1IS | Timer 1 Interrupt Status Bit<br>0: Timer 1 has not matched the conditions configured in the FLXAnFRT1C register<br>1: Timer 1 matched the conditions configured in the FLXAnFRT1C register |
| 16 | T0IS | Timer 0 Interrupt Status Bit<br>0: Timer 0 has not matched the conditions configured in the FLXAnFRT0C register<br>1: Timer 0 matched the conditions configured in the FLXAnFRT0C register |
| 15 to 1 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 0 | OS | Operation Status Bit<br>0: Operation disabled, reset state<br>1: Operation enabled |

**(1) FLXAnFROS.T2IS**

Timer 2 Interrupt Status Bit

Writing 0 has no effect on the bit value.

This bit represents that the expiration criteria configured in FLXAnFRT2C register has matched the FlexRay local time.

If enabled in FLXAnFROC.T2IE the timer 2 interrupt is generated when FLXAnFROS.T2IS is 1.

[Clearing condition]

This bit is cleared by writing 1 to FLXAnFROS.T2IS.
This bit is cleared when FLXAnFROS.OS changes from 1 to 0.

[Setting condition]

This bit is set when the expiration criteria configured in FLXAnFRT2C matches the FlexRay local time.

**(2) FLXAnFROS.T1IS**

Timer 1 Interrupt Status Bit

Writing 0 has no effect on the bit value.

This bit represents that the expiration criteria configured in FLXAnFRT1C register has matched the FlexRay local time.

If enabled in FLXAnFROC.T1IE the timer 1 interrupt is generated when FLXAnFROS.T1IS is 1.

[Clearing condition]

This bit is cleared by writing 1 to FLXAnFROS.T1IS.
This bit is cleared when FLXAnFROS.OS changes from 1 to 0.

[Setting condition]

This bit is set when the expiration criteria configured in FLXAnFRT1C matches the FlexRay local time.

**(3) FLXAnFROS.T0IS**

Timer 0 Interrupt Status Bit

Writing 0 has no effect on the bit value.

This bit represents that the expiration criteria configured in FLXAnFRT0C register has matched the FlexRay local time.

If enabled in FLXAnFROC.T0IE the timer 0 interrupt is generated when FLXAnFROS.T0IS is 1.

[Clearing condition]

This bit is cleared by writing 1 to FLXAnFROS.T0IS.
This bit is cleared when FLXAnFROS.OS changes from 1 to 0.

[Setting condition]

This bit is set when the expiration criteria configured in FLXAnFRT0C matches the FlexRay local time.

### (4) FLXAnFROS.OS

Operation Status Bit

This bit represents if the FlexRay module is in the reset or the operation state.

When FLXAnFROS.OS is 0 the FlexRay module gets initialized and registers mapped to the address area <FLXn_base> + $0010_H$ to <FLXn_base> + $0FFF_H$ cannot be accessed; read access from these registers will return undefined data.

When FLXAnFROS.OS is 1 it is possible to access to the address area <FLXn_base> + $0010_H$ to <FLXn_base> + $0FFF_H$ and to perform FlexRay communication.

When FLXAnFROS.OS changes from 0 to 1 all registers in the address range <FLXn_base> + $0010_H$ to <FLXn_base> + $0FFF_H$ are set to the “Values after reset”.

[Clearing condition]

When FLXAnFROC.OE is set to 0. It takes up to two peripheral bus clock cycles until FLXAnFROS.OS is set to 0.

[Setting condition]

When FLXAnFROC.OE is set to 1 it takes up to four peripheral clock cycles of the clock with the lower frequency out of the FlexRay sample clock and peripheral bus clock until FLXAnFROS.OS is set to 1.

## 20.2.3 Special Registers

### 20.2.3.1 FLXAnFRLCK — FlexRay Lock Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXAn_base> + $001C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CLK[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.10 FLXAnFRLCK Register Contents**

| Bit | Symbol | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 7 to 0 | CLK[7:0] | Configuration Lock Key Bit |

#### (1) FLXAnFRLCK.CLK

Configuration Lock Key Bit

The Lock Register is write-only. Reading the register will return $0000\ 0000_H$.

To leave CONFIG state by writing FLXAnFRSUCC1.CMD (command READY), the write operation has to be directly preceded by two write accesses to the Configuration Lock Key (unlock sequence). If the write sequence below is interrupted by other write accesses between the second write to the Configuration Lock Key and the write access to the FLXAnFRSUCC1 register, the CC remains in CONFIG state and the sequence has to be repeated.

First write: FLXAnFRLCK.CLK = "$1100\ 1110_B$" (0xCE)

Second write: FLXAnFRLCK.CLK = "$0011\ 0001_B$" (0x31)

Third write: FLXAnFRSUCC1.CMD

**CAUTION**

In case that the Host uses 8/16-bit accesses to write the listed bit fields, the programmer has to ensure that no "dummy accesses" e.g. to the remaining register bytes / words are inserted by the compiler.

## 20.2.4 Interrupt Registers

### 20.2.4.1 FLXAnFREIR — FlexRay Error Interrupt Register

The flags are set when the CC detects one of the listed error conditions. The flags remain set until the Host clears them.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0020_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TABB | LTVB | EDB | — | — | — | — | — | TABA | LTVA | EDA |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MHF | IOBA | IIBA | EFA | RFO | AERR | CCL | CCF | SFO | SFBM | CNA | PEMC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.11 FLXAnFREIR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 26 | TABB | Transmission Across Boundary Channel B Flag<br>0: No transmission across slot boundary detected on channel B<br>1: Transmission across slot boundary detected on channel B |
| 25 | LTVB | Latest Transmit Violation Channel B Flag<br>0: No latest transmit violation detected on channel B<br>1: Latest transmit violation detected on channel B |
| 24 | EDB | Error Detected on Channel B Flag<br>0: No error detected on channel B<br>1: Error detected on channel B |
| 23 to 19 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 18 | TABA | Transmission Across Boundary Channel A Flag<br>0: No transmission across slot boundary detected on channel A<br>1: Transmission across slot boundary detected on channel A |
| 17 | LTVA | Latest Transmit Violation Channel A Flag<br>0: No latest transmit violation detected on channel A<br>1: Latest transmit violation detected on channel A |
| 16 | EDA | Error Detected on Channel A Flag<br>0: No error detected on channel A<br>1: Error detected on channel A |
| 15 to 12 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 11 | MHF | Message Handler Constraints Flag<br>0: No Message Handler failure detected<br>1: Message Handler failure detected |
| 10 | IOBA | Illegal Output buffer Access Flag<br>0: No illegal Host access to Output Buffer occurred<br>1: Illegal Host access to Output Buffer occurred |

Table 20.11 FLXAnFREIR Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 9 | IIBA | Illegal Input Buffer Access Flag<br>0: No illegal Host access to Input Buffer occurred<br>1: Illegal Host access to Input Buffer occurred |
| 8 | EFA | Empty FIFO Access Flag<br>0: No Host access to empty FIFO occurred<br>1: Host access to empty FIFO occurred |
| 7 | RFO | Receive FIFO Overrun Flag<br>0: No receive FIFO overrun detected<br>1: A receive FIFO overrun has been detected |
| 6 | AERR | Access error flag Flag<br>0: Access error is not detected.<br>1: Access error is detected. |
| 5 | CCL | CHI Command Locked Flag<br>0: CHI command accepted<br>1: CHI command not accepted |
| 4 | CCF | Clock Correction Failure Flag<br>0: No clock correction error<br>1: Clock correction failed |
| 3 | SFO | Sync Frame Overflow Flag<br>0: Number of received sync frames ≤ FLXAnFRGTUC2.SNM<br>1: More sync frames received than configured by FLXAnFRGTUC2.SNM |
| 2 | SFBM | Sync Frames Below Minimum Flag<br>0: Sync node: 1 or more sync frames received, non-sync node: 2 or more sync frames received<br>1: Less than the required minimum of sync frames received |
| 1 | CNA | Command Not Accepted Flag<br>0: CHI command accepted<br>1: CHI command not accepted |
| 0 | PEMC | POC Error Mode Changed Flag<br>0: Error mode has not changed<br>1: Error mode has changed |

### (1) FLXAnFREIR.TABB

Transmission Across Boundary Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

The flag signals to the Host that a transmission across a slot boundary occurred for channel B.

### (2) FLXAnFREIR.LTVB

Latest Transmit Violation Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

The flag signals a latest transmit violation on channel B to the Host.

**(3) FLXAnFREIR.EDB**

Error Detected on Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This bit is set whenever one of the flags FLXAnFRACS.SEDB, FLXAnFRACS.CEDB, FLXAnFRACS.CIB, FLXAnFRACS.SBVB changes from 0 to 1.

**(4) FLXAnFREIR.TABA**

Transmission Across Boundary Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

The flag signals to the Host that a transmission across a slot boundary occurred for channel A.

**(5) FLXAnFREIR.LTVA**

Latest Transmit Violation Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

The flag signals a latest transmit violation on channel A to the Host.

**(6) FLXAnFREIR.EDA**

Error Detected on Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This bit is set whenever one of the flags FLXAnFRACS.SEDA, FLXAnFRACS.CEDA, FLXAnFRACS.CIA, FLXAnFRACS.SBVA changes from 0 to 1.

**(7) FLXAnFREIR.MHF**

Message Handler Constraints Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

The flag signals a Message Handler constraints violation condition. It is set whenever one of the flags FLXAnFRMHDF.SNUA, FLXAnFRMHDF.SNUB, FLXAnFRMHDF.FNFA, FLXAnFRMHDF.FNFB, FLXAnFRMHDF.TBFA, FLXAnFRMHDF.TBFB, FLXAnFRMHDF.WAHP changes from 0 to 1.

**(8) FLXAnFREIR.IOBA**

Illegal Output buffer Access Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set by the CC when the Host requests the transfer of a message buffer from the Message RAM to the Output Buffer while FLXAnFROBCR.OBSYS is set to 1.

**(9) FLXAnFREIR.IIBA**

Illegal Input Buffer Access Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set by the CC when the Host wants to modify a message buffer via Input Buffer and one of the following conditions applies:

1. The CC is not in CONFIG or DEFAULT_CONFIG state and the Host writes to the Input Buffer Command Request register to modify the
   - Header section of message buffer 0, 1 if configured for transmission in key slot
   - Header section of static message buffers with buffer number < FLXAnFRMRC.FDB while FLXAnFRMRC.SEC = “01”
   - Header section of any static or dynamic message buffer while FLXAnFRMRC.SEC = “1x”
   - Header and / or data section of any message buffer belonging to the receive FIFO
2. The Host writes to any register of the Input Buffer while FLXAnFRIBCR.IBSYH is set to 1.

**(10) FLXAnFREIR.EFA**

Empty FIFO Access Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set by the CC when the Host requests the transfer of a message from the receive FIFO via Output Buffer while the receive FIFO is empty.

**(11) FLXAnFREIR.RFO**

Receive FIFO Overrun Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

The flag is set by the CC when a receive FIFO overrun is detected. When a receive FIFO overrun occurs, the oldest message is overwritten with the actual received message. The actual state of the FIFO is monitored in register FLXAnFRFSR.

**(12) FLXAnFREIR.AERR**

Access error flag Flag

Writing 0 in this bit has no effect.

This bit is cleared when writing 1 to it.

Notifies of an access error.

When the AMR, ATBF1, or ATBF2 bit in the FLXAnFRMHDS register changes from 0 to 1, this bit is set to 1.

**(13) FLXAnFREIR.CCL**

CHI Command Locked Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

The flag signals that the write access to the CHI command vector FLXAnFRSUCC1.CMD was not successful because the execution of the previous CHI command has not yet completed. In this case bit FLXAnFREIR.CNA is also set to 1.

**(14) FLXAnFREIR.CCF**

Clock Correction Failure Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set at the end of the communication cycle whenever one of the following errors occurred:

- Missing offset and / or rate correction
- Clock correction limit reached

The clock correction status is monitored in registers FLXAnFRCCEV and FLXAnFRSFS. A failure may occur during startup, therefore bit FLXAnFREIR.CCF should be set to 0 after the CC entered NORMAL_ACTIVE state.

**(15) FLXAnFREIR.SFO**

Sync Frame Overflow Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

It is set to 1 when either the number of sync frames received during the last communication cycle or the total number of different sync frame IDs received during the last double cycle exceeds the maximum number of sync frames as defined by FLXAnFRGTUC2.SNM.

**(16) FLXAnFREIR.SFBM**

Sync Frames Below Minimum Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set to 1 at the end of a cycle if the number of sync frames received during the last communication cycle was below the limit required for rate or offset correction term calculation (i.e. missing offset and / or missing rate correction). The clock correction status is monitored in FLXAnFRCCEV and FLXAnFRSFS.

This flag may be set to 1 during startup. Therefore this flag should be set to 0 by the Host after the CC entered NORMAL_ACTIVE state.

**(17) FLXAnFREIR.CNA**

Command Not Accepted Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

The flag signals that the write access to the CHI command vector FLXAnFRSUCC1.CMD was not successful because the requested command was not valid in the actual POC state, or because the CHI command was locked (FLXAnFREIR.CCL = 1).

**(18) FLXAnFREIR.PEMC**

POC Error Mode Changed Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set to 1 whenever the error mode signaled by FLXAnFRCCEV.ERRM has changed.

### 20.2.4.2 FLXAnFRSIR — FlexRay Status Interrupt Register

The flags are set when the CC detects one of the listed events. The flags remain set until the Host clears them.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0024_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | MTSB | WUPB | — | — | — | — | — | — | MTSA | WUPA |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SDS | MBSI | SUCS | SWE | TOBC | TIBC | TI1 | TI0 | NMVC | RFCL | RFNE | RXI | TXI | CYCS | CAS | WST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.12 FLXAnFRSIR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 25 | MTSB | MTS Received on Channel B Flag (vSS!ValidMTSB)<br>0: No MTS symbol received on channel B<br>1: MTS symbol received on channel B |
| 24 | WUPB | Wakeup Pattern Channel B Flag<br>0: No wakeup pattern received on channel B<br>1: Wakeup pattern received on channel B |
| 23 to 18 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 17 | MTSA | MTS Received on Channel A Flag (vSS!ValidMTSA)<br>0: No MTS symbol received on channel A<br>1: MTS symbol received on channel A |
| 16 | WUPA | Wakeup Pattern Channel A Flag<br>0: No wakeup pattern received on channel A<br>1: Wakeup pattern received on channel A |
| 15 | SDS | Start of Dynamic Segment Flag<br>0: Dynamic segment not yet started<br>1: Dynamic segment started |
| 14 | MBSI | Message Buffer Status Interrupt Flag<br>0: No message buffer status change of message buffer with MBI = 1<br>1: Message buffer status of at least one message buffer with MBI = 1 has changed |
| 13 | SUCS | Startup Completed Successfully Flag<br>0: No startup completed successfully<br>1: Startup completed successfully |
| 12 | SWE | Stop Watch Event Flag<br>0: No Stop Watch Event<br>1: Stop Watch Event occurred |
| 11 | TOBC | Transfer Output Buffer Completed Flag<br>0: No transfer completed<br>1: Transfer between Message RAM and Output Buffer completed |

Table 20.12 FLXAnFRSIR Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 10 | TIBC | Transfer Input Buffer Completed Flag<br>0: No transfer completed<br>1: Transfer between Input Buffer and Message RAM completed |
| 9 | TI1 | Timer Interrupt 1 Flag<br>0: No timer interrupt 1<br>1: Timer interrupt 1 occurred |
| 8 | TI0 | Timer Interrupt 0 Flag<br>0: No timer interrupt 0<br>1: Timer interrupt 0 occurred |
| 7 | NMVC | Network Management Vector Changed Flag<br>0: No change in the network management vector<br>1: Network management vector changed |
| 6 | RFCL | Receive FIFO Critical Level Flag<br>0: Receive FIFO below critical level<br>1: Receive FIFO critical level reached |
| 5 | RFNE | Receive FIFO Not Empty Flag<br>0: Receive FIFO is empty<br>1: Receive FIFO is not empty |
| 4 | RXI | Receive Interrupt Flag<br>0: No ND flag of a receive buffer with MBI = 1 has been set to 1<br>1: At least one ND flag of a receive buffer with MBI = 1 has been set to 1 |
| 3 | TXI | Transmit Interrupt Flag<br>0: No frame transmitted from a transmit buffer with MBI = 1<br>1: At least one frame was transmitted from a transmit buffer with MBI = 1 |
| 2 | CYCS | Cycle Start Interrupt Flag<br>0: No communication cycle started<br>1: Communication cycle started |
| 1 | CAS | Collision Avoidance Symbol Flag<br>0: No bit pattern matching the CAS symbol received<br>1: Bit pattern matching the CAS symbol received |
| 0 | WST | Wakeup Status Flag<br>0: Wakeup status unchanged<br>1: Wakeup status changed |

### (1) FLXAnFRSIR.MTSB

MTS Received on Channel B Flag (vSS!ValidMTSB)

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

Media Access Test symbol received on channel B during the preceding symbol window.

Updated by the CC for each channel at the end of the symbol window.

### (2) FLXAnFRSIR.WUPB

Wakeup Pattern Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set to 1 when a wakeup pattern was received on channel B in either of the following states:

- WAKEUP
- READY
- STARTUP state

**(3) FLXAnFRSIR.MTSA**

MTS Received on Channel A Flag (vSS!ValidMTSA)

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

Media Access Test symbol received on channel A during the preceding symbol window.

Updated by the CC for each channel at the end of the symbol window.

**(4) FLXAnFRSIR.WUPA**

Wakeup Pattern Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set to 1 when a wakeup pattern was received on channel A in either of the following states:

- WAKEUP
- READY
- STARTUP state

**(5) FLXAnFRSIR.SDS**

Start of Dynamic Segment Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set by the CC when the dynamic segment starts.

**(6) FLXAnFRSIR.MBSI**

Message Buffer Status Interrupt Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set by the CC when the message buffer status FLXAnFRMBS has changed and if bit MBI of that message buffer is 1 (see **Table 20.120**).

**(7) FLXAnFRSIR.SUCS**

Startup Completed Successfully Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set whenever a startup completed successfully and the CC entered NORMAL_ACTIVE state.

**(8) FLXAnFRSIR.SWE**

Stop Watch Event Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set after a stop watch activation when the actual cycle counter and macrotick value are stored in the Stop Watch register (see FLXAnFRSTPW1 — FlexRay Stop Watch Register 1 in **Section 20.2.5.4, FLXAnFRSTPW1 — FlexRay Stop Watch Register 1**).

**(9) FLXAnFRSIR.TOBC**

Transfer Output Buffer Completed Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set whenever a transfer from the Message RAM to the Output Buffer has completed and FLXAnFROBCR.OBSYS has been reset by the Message Handler.

**(10) FLXAnFRSIR.TIBC**

Transfer Input Buffer Completed Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set whenever a transfer from Input Buffer to the Message RAM has completed and FLXAnFRIBCR.IBSYS has been reset by the Message Handler.

**(11) FLXAnFRSIR.TI1**

Timer 1 Interrupt Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set whenever timer 1 matches the conditions configured in register FLXAnFRT1C.

FlexRay timer 1 interrupt is generated when the T1IE bit in the FLXAnFROC register is effective.

**(12) FLXAnFRSIR.TI0**

Timer 0 Interrupt Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set whenever timer 0 matches the conditions configured in register FLXAnFRT0C.

FlexRay timer 0 interrupt is generated when the T0IE bit in the FLXAnFROC register is effective.

**(13) FLXAnFRSIR.NMVC**

Network Management Vector Changed Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This is set when a change in the Network Management Vector occurs.

**(14) FLXAnFRSIR.RFCL**

Receive FIFO Critical Level Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set when the receive FIFO fill level FLXAnFRFSR.RFFL is equal or greater than the critical level as configured by FLXAnFRFCL.CL.

**(15) FLXAnFRSIR.RFNE**

Receive FIFO Not Empty Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set by the CC when a received valid frame was stored into the empty receive FIFO. The actual state of the receive FIFO is monitored in register FLXAnFRFSR.

**(16) FLXAnFRSIR.RXI**

Receive Interrupt Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set by the CC whenever the set condition of a message buffers ND flag is fulfilled (see **Section 20.2.9.6, FLXAnFRNDATi — FlexRay New Data Register i (i = 1 to 4)**), and if bit MBI of that message buffer is set to 1 (see **Table 20.120**)

**(17) FLXAnFRSIR.TXI**

Transmit Interrupt Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set by the CC at the end of frame transmission if bit MBI in the respective message buffer is set to 1 (see **Table 20.120**).

**(18) FLXAnFRSIR.CYCS**

Cycle Start Interrupt Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set by the CC when a communication cycle starts.

**(19) FLXAnFRSIR.CAS**

Collision Avoidance Symbol Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set by the CC during STARTUP state when a CAS or a potential CAS was received.

**(20) FLXAnFRSIR.WST**

Wakeup Status Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set when FLXAnFRCCSV.WSV changes to a value other than UNDEFINED.

### 20.2.4.3 FLXAnFREILS — FlexRay Error Interrupt Line Select Register

The FlexRay Error Interrupt Line Select register assigns an interrupt generated by a specific error interrupt flag from register FLXAnFREIR to one of the two modules interrupt lines (FlexRay 0 interrupt, FlexRay 1 interrupt).

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0028_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TABBL | LTVBL | EDBL | — | — | — | — | — | TABAL | LTVAL | EDAL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MHFL | IOBAL | IIBAL | EFAL | RFOL | AERRL | CCLL | CCFL | SFOL | SFBML | CNAL | PEMCL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.13 FLXAnFREILS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 26 | TABBL | Transmission Across Boundary Channel B Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 25 | LTVBL | Latest Transmit Violation Channel B Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 24 | EDBL | Error Detected on Channel B Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 23 to 19 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 18 | TABAL | Transmission Across Boundary Channel A Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 17 | LTVAL | Latest Transmit Violation Channel A Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 16 | EDAL | Error Detected on Channel A Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 15 to 12 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 11 | MHFL | Message Handler Constraints Flag Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 10 | IOBAL | Illegal Output Buffer Access Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |

**Table 20.13 FLXAnFREILS Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 9 | IIBAL | Illegal Input Buffer Access Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 8 | EFAL | Empty FIFO Access Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 7 | RFOL | Receive FIFO Overrun Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 6 | AERRL | Access Error Interrupt Output Select Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 5 | CCLL | CHI Command Locked Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 4 | CCFL | Clock Correction Failure Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 3 | SFOL | Sync Frame Overflow Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 2 | SFBML | Sync Frames Below Minimum Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 1 | CNAL | Command Not Accepted Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 0 | PEMCL | POC Error Mode Changed Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |

### 20.2.4.4 FLXAnFRSILS — FlexRay Status Interrupt Line Select Register

The FlexRay Status Interrupt Line Select register assigns an interrupt generated by a specific status interrupt flag from register FLXAnFRSIR to one of the two module interrupt lines (FlexRay 0 interrupt, FlexRay 1 interrupt).

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $002C_H$

**Value after reset:** $0303\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | MTSBL | WUPBL | — | — | — | — | — | — | MTSAL | WUPAL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SDSL | MBSIL | SUCSL | SWEL | TOBCL | TIBCL | TI1L | TI0L | NMVCL | RFCLL | RFNEL | RXIL | TXIL | CYCSL | CASL | WSTL |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.14 FLXAnFRSILS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 25 | MTSBL | Media Access Test Symbol Channel B Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 24 | WUPBL | Wakeup Pattern Channel B Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 23 to 18 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 17 | MTSAL | Media Access Test Symbol Channel A Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 16 | WUPAL | Wakeup Pattern Channel A Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 15 | SDSL | Start of Dynamic Segment Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 14 | MBSIL | Message Buffer Status Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 13 | SUCSL | Startup Completed Successfully Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 12 | SWEL | Stop Watch Event Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 11 | TOBCL | Transfer Output Buffer Completed Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |

Table 20.14 FLXAnFRSILS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 10 | TIBCL | Transfer Input Buffer Completed Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 9 | TI1L | Timer 1 Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 8 | TI0L | Timer 0 Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 7 | NMVCL | Network Management Vector Changed Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 6 | RFCLL | Receive FIFO Critical Level Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 5 | RFNEL | Receive FIFO Not Empty Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 4 | RXIL | Receive Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 3 | TXIL | Transmit Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 2 | CYCSL | Cycle Start Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 1 | CASL | Collision Avoidance Symbol Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |
| 0 | WSTL | Wakeup Status Interrupt Line Bit<br>0: Interrupt assigned to FlexRay 0 interrupt<br>1: Interrupt assigned to FlexRay 1 interrupt |

### 20.2.4.5 FLXAnFREIES — FlexRay Error Interrupt Enable Set Register

The settings in the FlexRay Error Interrupt Enable Set (FLXAnFREIES) and FlexRay Error Interrupt Enable Reset (FLXAnFREIER) register determine which status changes in the FlexRay Error Interrupt Register will result in an interrupt.

The enable bits are set by writing to FLXAnFREIES and reset by writing to FLXAnFREIER. Reading from both addresses will result in the same value.

Writing 0 has no effect on the bit value.

Writing a 1 sets the interrupt enable bit.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0030_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TABBE | LTVBE | EDBE | — | — | — | — | — | TABAE | LTVAE | EDAE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MHFE | IOBAE | IIBAE | EFAE | RFOE | AERRE | CCLE | CCFE | SFOE | SFBME | CNAE | PEMCE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.15 FLXAnFREIES Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 26 | TABBE | Transmission Across Boundary Channel B Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 25 | LTVBE | Latest Transmit Violation Channel B Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 24 | EDBE | Error Detected on Channel B Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 23 to 19 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 18 | TABAE | Transmission Across Boundary Channel A Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 17 | LTVAE | Latest Transmit Violation Channel A Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 16 | EDAE | Error Detected on Channel A Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 15 to 12 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |

Table 20.15 FLXAnFREIES Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | MHFE | Message Handler Constraints Flag Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 10 | IOBAE | Illegal Output Buffer Access Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 9 | IIBAE | Illegal Input Buffer Access Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 8 | EFAE | Empty FIFO Access Interrupt Enable3 Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 7 | RFOE | Receive FIFO Overrun Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 6 | AERRE | Access Error Interrupt Enable Bit<br>0: Interrupt is disabled.<br>1: Interrupt is enabled. |
| 5 | CCLE | CHI Command Locked Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 4 | CCFE | Clock Correction Failure Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 3 | SFOE | Sync Frame Overflow Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 2 | SFBME | Sync Frames Below Minimum Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 1 | CNAE | Command Not Accepted Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 0 | PEMCE | POC Error Mode Changed Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |

### 20.2.4.6 FLXAnFREIER — FlexRay Error Interrupt Enable Reset Register

The settings in the FlexRay Error Interrupt Enable Set (FLXAnFREIES) and FlexRay Error Interrupt Enable Reset (FLXAnFREIER) register determine which status changes in the FlexRay Error Interrupt Register will result in an interrupt.

The enable bits are set by writing to FLXAnFREIES and reset by writing to FLXAnFREIER. Reading from both addresses will result in the same value.

Writing 0 has no effect on the bit value.

Writing a 1 clears the interrupt enable bit.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0034_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TABBD | LTVBD | EDBD | — | — | — | — | — | TABAD | LTVAD | EDAD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MHFD | IOBAD | IIBAD | EFAD | RFOD | AERRD | CCLD | CCFD | SFOD | SFBMD | CNAD | PEMCD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.16 FLXAnFREIER Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 26 | TABBD | Transmission Across Boundary Channel B Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 25 | LTVBD | Latest Transmit Violation Channel B Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 24 | EDBD | Error Detected on Channel B Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 23 to 19 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 18 | TABAD | Transmission Across Boundary Channel A Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 17 | LTVAD | Latest Transmit Violation Channel A Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 16 | EDAD | Error Detected on Channel A Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 15 to 12 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |

Table 20.16 FLXAnFREIER Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | MHFD | Message Handler Constraints Flag Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 10 | IOBAD | Illegal Output Buffer Access Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 9 | IIBAD | Illegal Input Buffer Access Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 8 | EFAD | Empty FIFO Access Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 7 | RFOD | Receive FIFO Overrun Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 6 | AERRD | Access error interrupt disable Bit<br>0: Interrupt is disabled.<br>1: Interrupt is enabled. |
| 5 | CCLD | CHI Command Locked Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 4 | CCFD | Clock Correction Failure Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 3 | SFOD | Sync Frame Overflow Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 2 | SFBMD | Sync Frames Below Minimum Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 1 | CNAD | Command Not Accepted Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 0 | PEMCD | POC Error Mode Changed Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |

### 20.2.4.7 FLXAnFRSIES — FlexRay Status Interrupt Enable Set Register

The settings in the FlexRay Status Interrupt Enable Set (FLXAnFRSIES) and FlexRay Status Interrupt Enable Reset (FLXAnFRSIER) register determine which status changes in the FlexRay Status Interrupt Register will result in an interrupt.

The enable bits are set by writing to FLXAnFRSIES and reset by writing to FLXAnFRSIER. Reading from both addresses will result in the same value.

Writing 0 has no effect on the bit value.

Writing a 1 sets the interrupt enable bit.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0038_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | MTSBE | WUPBE | — | — | — | — | — | — | MTSAE | WUPAE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SDSE | MBSIE | SUCSE | SWEE | TOBCE | TIBCE | TI1E | TI0E | NMVCE | RFCLE | RFNEE | RXIE | TXIE | CYCSE | CASE | WSTE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.17 FLXAnFRSIES Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 25 | MTSBE | MTS Received on Channel B Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 24 | WUPBE | Wakeup Pattern Channel B Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 23 to 18 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 17 | MTSAE | MTS Received on Channel A Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 16 | WUPAE | Wakeup Pattern Channel A Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 15 | SDSE | Start of Dynamic Segment Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 14 | MBSIE | Message Buffer Status Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 13 | SUCSE | Startup Completed Successfully Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |

Table 20.17 FLXAnFRSIES Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 12 | SWEE | Stop Watch Event Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 11 | TOBCE | Transfer Output Buffer Completed Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 10 | TIBCE | Transfer Input Buffer Completed Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 9 | TI1E | Timer 1 Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 8 | TI0E | Timer 0 Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 7 | NMVCE | Network Management Vector Changed Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 6 | RFCLE | Receive FIFO Critical Level Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 5 | RFNEE | Receive FIFO Not Empty Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 4 | RXIE | Receive Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 3 | TXIE | Transmit Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 2 | CYCSE | Cycle Start Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 1 | CASE | Collision Avoidance Symbol Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 0 | WSTE | Wakeup Status Interrupt Enable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |

### 20.2.4.8 FLXAnFRSIER — FlexRay Status Interrupt Enable Reset Register

The settings in the FlexRay Status Interrupt Enable Set (FLXAnFRSIES) and FlexRay Status Interrupt Enable Reset (FLXAnFRSIER) register determine which status changes in the FlexRay Status Interrupt Register will result in an interrupt.

The enable bits are set by writing to FLXAnFRSIES and reset by writing to FLXAnFRSIER. Reading from both addresses will result in the same value.

Writing 0 has no effect on the bit value.

Writing a 1 clears the interrupt enable bit.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $003C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | MTSBD | WUPBD | — | — | — | — | — | — | MTSAD | WUPAD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SDSD | MBSID | SUCSD | SWED | TOBCD | TIBCD | TI1D | TI0D | NMVCD | RFCLD | RFNED | RXID | TXID | CYCSD | CASD | WSTD |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.18 FLXAnFRSIER Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 25 | MTSBD | MTS Received on Channel B Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 24 | WUPBD | Wakeup Pattern Channel B Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 23 to 18 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 17 | MTSAD | MTS Received on Channel A Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 16 | WUPAD | Wakeup Pattern Channel A Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 15 | SDSD | Start of Dynamic Segment Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 14 | MBSID | Message Buffer Status Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 13 | SUCSD | Startup Completed Successfully Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |

Table 20.18 FLXAnFRSIER Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 12 | SWED | Stop Watch Event Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 11 | TOBCD | Transfer Output Buffer Completed Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 10 | TIBCD | Transfer Input Buffer Completed Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 9 | TI1D | Timer Interrupt 1 Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 8 | TI0D | Timer Interrupt 0 Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 7 | NMVCD | Network Management Vector Changed Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 6 | RFCLD | Receive FIFO Critical Level Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 5 | RFNED | Receive FIFO Not Empty Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 4 | RXID | Receive Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 3 | TXID | Transmit Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 2 | CYCSD | Cycle Start Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 1 | CASD | Collision Avoidance Symbol Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 0 | WSTD | Wakeup Status Interrupt Disable Bit<br>0: Interrupt disabled<br>1: Interrupt enabled |

### 20.2.4.9 FLXAnFRILE — FlexRay Interrupt Line Enable Register

Each of the two module interrupt lines (FlexRay 0 interrupt, FlexRay 1 interrupt) can be enabled / disabled separately by programming bit FLXAnFRILE.EINT0 and FLXAnFRILE.EINT1.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0040_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EINT1 | EINT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 20.19 FLXAnFRILE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | EINT1 | Enable FlexRay 1 Interrupt Line Bit<br>0: FlexRay 1 interrupt disabled<br>1: FlexRay 1 interrupt enabled |
| 0 | EINT0 | Enable FlexRay 0 Interrupt Line Bit<br>0: FlexRay 0 interrupt disabled<br>1: FlexRay 0 interrupt enabled |

## 20.2.5 FlexRay Timer Registers

### 20.2.5.1 FLXAnFRT0C — FlexRay Timer 0 Configuration Register

This register is an absolute timer. It specifies the point in time when a FlexRay timer 0 interrupt occurs as the values of cycle count and macrotick (MT). When the FlexRay timer 0 passes, FLXAnFRSIR.TI0 and FLXAnFROS.T0IS are set to 1. A timer 0 interrupt then occurs while the FLXAnFROC.T0IE bit is effective.

**CAUTION**

The configuration of timer 0 is compared against the macrotick counter value, there is no separate counter for timer 0.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0044_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | T0MO[13:0] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | T0CC[6:0] | | | | | | | — | — | — | — | — | — | T0MS | T0RC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

**Table 20.20 FLXAnFRT0C Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 to 16 | T0MO[13:0] | Timer 0 Macrotick Offset Bit<br>Timer 0 Macrotick Offset |
| 15 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 14 to 8 | T0CC[6:0] | Timer 0 Cycle Code Bit<br>Timer 0 Cycle Code |
| 7 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | T0MS | Timer 0 Mode Select Bit<br>0: Single-shot mode<br>1: Continuous mode |
| 0 | T0RC | Timer 0 Run Control Bit<br>0: Timer 0 halted<br>1: Timer 0 running |

**(1) FLXAnFRT0C.T0MO**

Timer 0 Macrotick Offset Bit

Before reconfiguration of the timer, the timer has to be halted first by writing bit FLXAnFRT0C.T0RC to 0.

Configures the macrotick offset from the beginning of the communication cycle where the interrupt is to occur. The FlexRay timer 0 interrupt occurs at this offset for each cycle of the cycle set.

**(2) FLXAnFRT0C.T0CC**

Timer 0 Cycle Code Bit

Before reconfiguration of the timer, the timer has to be halted first by writing bit FLXAnFRT0C.T0RC to 0.

The 7-bit timer 0 cycle code determines the cycle set used for generation of the FlexRay timer 0 interrupt. For details about the configuration of the cycle code see **Section 20.3.8.2, Cycle Counter Filtering**.

**(3) FLXAnFRT0C.T0MS**

Timer 0 Mode Select Bit

Before reconfiguration of the timer, the timer has to be halted first by writing bit FLXAnFRT0C.T0RC to 0.

Configures the timer run mode. In Single-shot mode the timer is deactivated when the timer configuration matches the configured cycle counter and macrotick value.

**(4) FLXAnFRT0C.T0RC**

Timer 0 Run Control Bit

Timer 0 can be activated (set FLXAnFRT0C.T0RC to 1) when the POC is either in NORMAL_ACTIVE state or in NORMAL_PASSIVE state.

Timer 0 is deactivated when leaving NORMAL_ACTIVE state or NORMAL_PASSIVE state except for transitions between the two states.

### 20.2.5.2 FLXAnFRT1C — FlexRay Timer 1 Configuration Register

This register is a relative timer. After the specified number of macroticks (MT) has expired, a FlexRay timer 1 interrupt is asserted. When the FlexRay timer 1 passes, FLXAnFRSIR.TI1 and FLXAnFROS.T1IS are set to 1. A timer 1 interrupt then occurs while the FLXAnFROC.T1IE bit is effective.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0048_H$

**Value after reset:** $0002\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | T1MC[13:0] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | T1MS | T1RC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 20.21 FLXAnFRT1C Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 to 16 | T1MC[13:0] | Timer 1 Macrotick Count Bit |
| 15 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | T1MS | Timer 1 Mode Select Bit<br>0: Single-shot mode<br>1: Continuous mode |
| 0 | T1RC | Timer 1 Run Control Bit<br>0: Timer 1 halted<br>1: Timer 1 running |

**(1) FLXAnFRT1C.T1MC**

Timer 1 Macrotick Count Bit

Before reconfiguration of the timer, the timer has to be halted first by writing bit FLXAnFRT1C.T1RC to 0.

Valid values are 2 to 16383 MT in continuous mode

Valid values are 1 to 16383 MT in single-shot mode

When the configured macrotick count is reached the FlexRay timer 1 interrupt is generated.

**(2) FLXAnFRT1C.T1MS**

Timer 1 Mode Select Bit

Before reconfiguration of the timer, the timer has to be halted first by writing bit FLXAnFRT1C.T1RC to 0.

Configures the timer run mode. In Single-shot mode the timer is deactivated when the timer configuration matches the configured cycle counter and macrotick value.

**(3) FLXAnFRT1C.T1RC**

Timer 1 Run Control Bit

Timer 1 can be activated (set FLXAnFRT1C.T1RC to 1) as long as the POC is either in NORMAL_ACTIVE state or in NORMAL_PASSIVE state.

Timer 1 is deactivated when leaving NORMAL_ACTIVE state or NORMAL_PASSIVE state except for transitions between the two states.

### 20.2.5.3 FLXAnFRT2C — FlexRay Timer 2 Configuration Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0844_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | T2MO[13:0] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | T2CC[6:0] | | | | | | | — | — | — | — | — | — | T2MS | T2RC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

**Table 20.22 FLXAnFRT2C Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 to 16 | T2MO[13:0] | Timer 2 Macrotick Offset Bit<br>Timer 2 Macrotick Offset |
| 15 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 14 to 8 | T2CC[6:0] | Timer 2 Cycle Code Bit<br>Timer 2 Cycle Code |
| 7 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | T2MS | Timer 2 Mode Select Bit<br>0: Single-shot mode<br>1: Continuous mode |
| 0 | T2RC | Timer 2 Run Control Bit<br>0: Timer halted<br>1: Timer running |

**(1) FLXAnFRT2C.T2MO**

Timer 2 Macrotick Offset Bit

The user can only write to these bits when FLXAnFRT2C.T2RC is 0.

Allowed range is 0 to FLXAnFRGTUC2.MPC.

These bits represent the timer macrotick value defining the timer expiration condition.

**(2) FLXAnFRT2C.T2CC**

Timer 2 Cycle Code Bit

The user can only write to these bits when FLXAnFRT2C.T2RC is 0.

These bits represent the cycle counter filter code defining the timer expiration condition. See **Section 20.3.8.2, Cycle Counter Filtering** for cycle filtering.

**(3) FLXAnFRT2C.T2MS**

Timer 2 Mode Select Bit

The user can only write to these bits when FLXAnFRT2C.T2RC is 0.

This bit represents the operation mode of the Timer 2.

0: Single-shot mode

The timer is operating in the non-repetitive (single shot) mode. Once the configured expiration criteria are matching the timer will be automatically halted.

1: Continuous mode

The timer is operating in the repetitive (continuous) mode. The timer will expire every time the configured expiration criteria are matching. The timer is not halted.

**(4) FLXAnFRT2C.T2RC**

Timer 2 Run Control Bit

The user can only set this bit to 1 when the POC is in NORMAL_ACTIVE or NORMAL_PASSIVE state.

This bit represents the activation state of the Timer 2.

When the expiration criteria are matching for a single shot timer, then this bit is cleared automatically and the Timer 2 is halted.

[Setting condition]

This bit is set by writing 1 to it when the POC is in NORMAL_ACTIVE or NORMAL_PASSIVE state.

[Clearing condition]

This bit is cleared when the POC state leaves the NORMAL_ACTIVE or NORMAL_PASSIVE state except for transitions between the two states.
This bit is cleared when the timer is operating in the non-repetitive (single shot) mode (FLXAnFRT2C.T2MS is 0) and the expiration criteria are matching.
This bit is cleared by writing 0 to it.

### 20.2.5.4 FLXAnFRSTPW1 — FlexRay Stop Watch Register 1

The stop watch is activated by the following trigger events.

- FlexRay 0 interrupt or FlexRay 1 interrupt
- Writing bit FLXAnFRSTPW1.SSWT to 1

With the macrotick counter increment following next to the stop watch activation the actual cycle counter and macrotick values are captured in register FLXAnFRSTPW1 while the slot counter values for channel A and B are captured in register FLXAnFRSTPW2.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $004C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | SMTV[13:0] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | SCCV[5:0] | | | | | | — | EINT1 | EINT0 | — | SSWT | EDGE | SWMS | ESWT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R/W | R/W |

**Table 20.23 FLXAnFRSTPW1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 to 16 | SMTV[13:0] | Stop Watch Captured Macrotick Value |
| 15, 14 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 13 to 8 | SCCV[5:0] | Stop Watch Captured Cycle Counter Value |
| 7 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 6 | EINT1 | Enable FlexRay Interrupt 1 Trigger Bit<br>0: Stop watch trigger by FlexRay 1 interrupt disabled<br>1: FlexRay 1 interrupt event triggers stop watch |
| 5 | EINT0 | Enable FlexRay 0 Interrupt Trigger Bit<br>0: Stop watch trigger by FlexRay 0 interrupt disabled<br>1: FlexRay interrupt 0 event triggers stop watch |
| 4 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 3 | SSWT | Software Stop Watch Trigger Bit<br>0: Software trigger reset<br>1: Stop watch activated by software trigger |
| 2 | EDGE | Stop Watch Trigger Edge Select Bit<br>0: Falling edge<br>1: Rising edge |
| 1 | SWMS | Stop Watch Mode Select Bit<br>0: Single-shot mode<br>1: Continuous mode |

Table 20.23 FLXAnFRSTPW1 Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | ESWT | Enable Hardware Stop Watch Trigger Bit<br>0: Stop watch trigger disabled<br>1: Stop watch trigger enabled |

**(1) FLXAnFRSTPW1.SMTV**

Stop Watch Captured Macrotick Value

State of the macrotick counter when the stop watch event occurred.

**(2) FLXAnFRSTPW1.SCCV**

Stop Watch Captured Cycle Counter Value

State of the cycle counter when the stop watch event occurred.

**(3) FLXAnFRSTPW1.EINT1**

Enable FlexRay 1 Interrupt Trigger Bit

Enables stop watch trigger by FlexRay 1 interrupt when FLXAnFRSTPW1.ESWT = 1.

**(4) FLXAnFRSTPW1.EINT0**

Enable FlexRay 0 Interrupt Trigger Bit

Enables stop watch trigger by FlexRay 0 interrupt when FLXAnFRSTPW1.ESWT = 1.

**(5) FLXAnFRSTPW1.SSWT**

Software Stop Watch Trigger Bit

Bits FLXAnFRSTPW1.ESWT and FLXAnFRSTPW1.SSWT cannot be set to 1 simultaneously. In this case the write access to the register is ignored, and both bits keep their previous values. Either the external stop watch trigger or the software stop watch trigger may be used.

Writing 1 in this bit activates the stop watch. This bit is reset to 0 after the cycle count and slot count, and macrotick (MT) value are stored in the FlexRay stop watch register.

**(6) FLXAnFRSTPW1.EDGE**

Stop Watch Trigger Edge Select Bit

**(7) FLXAnFRSTPW1.SWMS**

Stop Watch Mode Select Bit

**(8) FLXAnFRSTPW1.ESWT**

Enable Stop Watch Trigger Bit

Bits FLXAnFRSTPW1.ESWT and FLXAnFRSTPW1.SSWT cannot be set to 1 simultaneously. In this case the write access to the register is ignored, and both bits keep their previous values. Either the external stop watch trigger or the software stop watch trigger may be used.

If enabled, a FlexRay 0 interrupt event or a FlexRay 1 interrupt event activates the stop watch.

In single-shot mode, this bit is reset to 0 after the cycle count and slot count, and macrotick (MT) value are stored in the FlexRay stop watch register.

### 20.2.5.5 FLXAnFRSTPW2 — FlexRay Stop Watch Register 2

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0050_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SSCVB[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SSCVA[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.24 FLXAnFRSTPW2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 26 to 16 | SSCVB[10:0] | Stop Watch Captured Slot Counter Value Channel B |
| 15 to 11 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 10 to 0 | SSCVA[10:0] | Stop Watch Captured Slot Counter Value Channel A |

#### (1) FLXAnFRSTPW2.SSCVB

Stop Watch Captured Slot Counter Value Channel B

State of the slot counter for channel B when the stop watch event occurred.

#### (2) FLXAnFRSTPW2.SSCVA

Stop Watch Captured Slot Counter Value Channel A

State of the slot counter for channel A when the stop watch event occurred.

## 20.2.6 CC Control Registers

This section describes the registers provided by the CC (Communication Controller) to allow the Host to control the operation of the CC. The FlexRay protocol specification requires the Host to write application configuration data in CONFIG state only. Please consider that the configuration registers are not locked for writing in DEFAULT_CONFIG state.

The configuration data is reset when DEFAULT_CONFIG state is entered from reset. To change POC state from DEFAULT_CONFIG to CONFIG state the Host has to apply CHI command CONFIG. If the Host wants the CC to leave CONFIG state, the Host has to execute the lock release sequence as described in **Section 20.2.3.1, FLXAnFRLCK — FlexRay Lock Register**.

### 20.2.6.1 FLXAnFRSUCC1 — FlexRay SUC Configuration Register 1

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0080_H$

**Value after reset:** $0C40\ 1080_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | CCHB | CCHA | MTSB | MTSA | HCSE | TSM | WUCS | PTA[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSA[4:0] | | | | | — | TXSY | TXST | PBSY | — | — | — | CMD[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 20.25 FLXAnFRSUCC1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 27 | CCHB | Connected to Channel B Bit<br>Configures pChannels<br>0: Not connected to channel B<br>1: Node connected to channel B (default after reset) |
| 26 | CCHA | Connected to Channel A Bit<br>Configures pChannels<br>0: Not connected to channel A<br>1: Node connected to channel A (default after reset) |
| 25 | MTSB | Select Channel B for MTS Transmission Bit<br>0: Channel B disabled for MTS transmission<br>1: Channel B selected for MTS transmission |
| 24 | MTSA | Select Channel A for MTS Transmission Bit<br>0: Channel A disabled for MTS transmission<br>1: Channel A selected for MTS transmission |
| 23 | HCSE | Halt due to Clock Sync Error Bit<br>Configures pAllowHaltDueToClock<br>0: CC will enter / remain in NORMAL_PASSIVE<br>1: CC will enter HALT state |
| 22 | TSM | Transmission Slot Mode Bit<br>Configures pSingleSlotEnabled<br>0: ALL Slot Mode<br>1: SINGLE Slot Mode (value after hard reset) |

**Table 20.25 FLXAnFRSUCC1 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 21 | WUCS | Wakeup Channel Select Bit<br>Configures pWakeupChannel<br>0: Send wakeup pattern on channel A<br>1: Send wakeup pattern on channel B |
| 20 to 16 | PTA[4:0] | Passive to Active Bit<br>Configures pAllowPassiveToActive |
| 15 to 11 | CSA[4:0] | Cold Start Attempts Bit<br>Configures gColdStartAttempts |
| 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 9 | TXSY | Transmit Sync Frame in Key Slot Bit<br>Configures pKeySlotUsedForSync<br>0: No sync frame transmission in key slot, node is neither sync nor coldstart node<br>1: Key slot used to transmit sync frame, node is sync node |
| 8 | TXST | Transmit Startup Frame in Key Slot Bit<br>Configures pKeySlotUsedForStartup<br>0: No startup frame transmission in key slot, node is non-coldstarter<br>1: Key slot used to transmit startup frame, node is leading or following coldstarter |
| 7 | PBSY | POC Busy Flag<br>0: POC not busy, FLXAnFRSUCC1.CMD writeable<br>1: POC is busy, FLXAnFRSUCC1.CMD locked |
| 6 to 4 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 3 to 0 | CMD[3:0] | CHI Command Vector Bit<br>0000: command_not_accepted<br>0001: CONFIG<br>0010: READY<br>0011: WAKEUP<br>0100: RUN<br>0101: ALL_SLOTS<br>0110: HALT<br>0111: FREEZE<br>1000: SEND_MTS<br>1001: ALLOW_COLDSTART<br>1010: RESET_STATUS_INDICATORS<br>1011: MONITOR_MODE<br>1100: CLEAR_RAMS<br>others: reserved |

**(1) FLXAnFRSUCC1.CCHB**

Connected to Channel B Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Configures whether the node is connected to channel B (pChannels).

**(2) FLXAnFRSUCC1.CCHA**

Connected to Channel A Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Configures whether the node is connected to channel A (pChannels).

**(3) FLXAnFRSUCC1.MTSB**

Select Channel B for MTS Transmission Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

FLXAnFRSUCC1.MTSB may also be changed outside DEFAULT_CONFIG or CONFIG state when the write to FLXAnFRSUCC1 register is directly preceded by the unlock sequence for the Configuration Lock Key as described in **Section 20.2.3.1, FLXAnFRLCK — FlexRay Lock Register**. This may be combined with CHI command SEND_MTS. If both bits FLXAnFRSUCC1.MTSA and FLXAnFRSUCC1.MTSB are set to 1 an MTS symbol will be transmitted on both channels when requested by writing FLXAnFRSUCC1.CMD = "$1000_B$".

The bit selects channel B for MTS symbol transmission.

**(4) FLXAnFRSUCC1.MTSA**

Select Channel A for MTS Transmission Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

FLXAnFRSUCC1.MTSA may also be changed outside DEFAULT_CONFIG or CONFIG state when the write to FLXAnFRSUCC1 register is directly preceded by the unlock sequence for the Configuration Lock Key as described in **Section 20.2.3.1, FLXAnFRLCK — FlexRay Lock Register**. This may be combined with CHI command SEND_MTS. If both bits FLXAnFRSUCC1.MTSA and FLXAnFRSUCC1.MTSB are set to 1 an MTS symbol will be transmitted on both channels when requested by writing FLXAnFRSUCC1.CMD = "$1000_B$".

The bit selects channel A for MTS symbol transmission.

**(5) FLXAnFRSUCC1.HCSE**

Halt due to Clock Sync Error Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Controls the transition to HALT state due to a clock synchronization error (pAllowHaltDueToClock).

**(6) FLXAnFRSUCC1.TSM**

Transmission Slot Mode Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Selects the value after transmission slot mode reset (pSingleSlotEnabled).

In SINGLE slot mode the CC may only transmit in the preconfigured key slot. The key slot ID is configured in the header section of message buffer 0 respectively message buffers 0 and 1 depending on bit FLXAnFRMRC.SPLM.

In case FLXAnFRSUCC1.TSM = 1, message buffer 0 respectively message buffers 0, 1 can be (re)configured in DEFAULT_CONFIG or CONFIG state only. In ALL slot mode the CC may transmit in all slots.

FLXAnFRSUCC1.TSM is a configuration bit which can only be set / reset by the Host.

The CC changes to ALL slot mode when the Host successfully applied the ALL_SLOTS command by writing FLXAnFRSUCC1.CMD = “0101” in POC states NORMAL_ACTIVE or NORMAL_PASSIVE. The actual slot mode is monitored by FLXAnFRCCSV.SLM.

**(7) FLXAnFRSUCC1.WUCS**

Wakeup Channel Select Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

With this bit the Host selects the channel on which the CC sends the Wakeup pattern (pWakeupChannel).

**(8) FLXAnFRSUCC1.PTA**

Passive to Active Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 31 even / odd cycle pairs.

Defines the number of consecutive even / odd cycle pairs that must have valid clock correction terms before the CC is allowed to transit from NORMAL_PASSIVE to NORMAL_ACTIVE state (pAllowPassiveToActive).

If set to “$00000_B$” the CC is not allowed to transit from NORMAL_PASSIVE to NORMAL_ACTIVE state.

**(9) FLXAnFRSUCC1.CSA**

Cold Start Attempts Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Must be identical in all nodes of a cluster.

Valid values are 2 to 31.

Configures the maximum number of attempts that a cold starting node is permitted to try to start up the network without receiving any valid response from another node (gColdStartAttempts).

### (10) FLXAnFRSUCC1.TXSY

Transmit Sync Frame in Key Slot Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Defines whether the key slot is used to transmit sync frames (pKeySlotUsedForSync).

**CAUTION**

The protocol requires that both bits FLXAnFRSUCC1.TXST and FLXAnFRSUCC1.TXSY are set for coldstart nodes.

### (11) FLXAnFRSUCC1.TXST

Transmit Startup Frame in Key Slot Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Defines whether the key slot is used to transmit startup frames (pKeySlotUsedForStartup).

**CAUTION**

The protocol requires that both bits FLXAnFRSUCC1.TXST and FLXAnFRSUCC1.TXSY are set for coldstart nodes.

### (12) FLXAnFRSUCC1.PBSY

POC Busy Flag

Signals that the POC is busy and cannot accept a command from the Host. FLXAnFRSUCC1.CMD is locked against write accesses.

Set to 1 after reset during initialization of internal RAM blocks.

### (13) FLXAnFRSUCC1.CMD

CHI Command Vector Bit

The Host may write any CHI command at any time, but certain commands are enabled only in certain POC states. If a command is not enabled, it will not be executed, the CHI command vector FLXAnFRSUCC1.CMD will be reset to "$0000_B$" = command_not_accepted, and flag FLXAnFREIR.CNA will be set to 1.

In general the Host must check FLXAnFRSUCC1.PBSY before writing a new CHI command.

In case the previous CHI command has not yet completed, FLXAnFREIR.CCL is set to 1 together with FLXAnFREIR.CNA; the CHI command needs to be repeated.

Except for HALT state, a POC state change command applied while the CC is already in the requested POC state neither causes a state change nor will FLXAnFREIR.CNA be set.

Reading FLXAnFRSUCC1.CMD shows whether the last CHI command was accepted. The actual POC state is monitored by FLXAnFRCCSV.POCS.

- command_not_accepted

  FLXAnFRSUCC1.CMD is reset to "$0000_B$" due to one of the following conditions:

  – Illegal command applied by the Host
  – Host applied command to leave CONFIG state without preceding config lock key
  – Host applied new command while execution of the previous Host command has not completed

– Host writes command_not_accepted

When FLXAnFRSUCC1.CMD is reset to "$0000_B$", FLXAnFREIR.CNA is set to 1, and if enabled an interrupt is generated.

Commands which are not accepted are not executed.

#### CONFIG

Go to POC state CONFIG when called in POC states DEFAULT_CONFIG, or READY. When called in HALT state the CC transits to POC state DEFAULT_CONFIG. When called in any other state, FLXAnFRSUCC1.CMD will be reset to "$0000_B$" = command_not_accepted.

#### READY

Go to POC state READY when called in POC states CONFIG, NORMAL_ACTIVE, NORMAL_PASSIVE, STARTUP, or WAKEUP. When called in any other state, FLXAnFRSUCC1.CMD will be reset to "$0000_B$" = command_not_accepted.

#### WAKEUP

Go to POC state WAKEUP when called in POC state READY. When called in any other state, FLXAnFRSUCC1.CMD will be reset to "$0000_B$" = command_not_accepted.

#### RUN

Go to POC state STARTUP when called in POC state READY. When called in any other state, FLXAnFRSUCC1.CMD will be reset to "$0000_B$" = command_not_accepted.

#### ALL_SLOTS

Leave SINGLE slot mode and go to ALL-SLOTS mode after successful startup / integration at the next end of cycle when called in POC states NORMAL_ACTIVE or NORMAL_PASSIVE. When called in any other state, FLXAnFRSUCC1.CMD will be reset to "$0000_B$" = command_not_accepted.

#### HALT

Set halt request FLXAnFRCCSV.HRQ to 1 and go to POC state HALT at the next end of cycle when called in POC states NORMAL_ACTIVE or NORMAL_PASSIVE. When called in any other state, FLXAnFRSUCC1.CMD will be reset to "$0000_B$" = command_not_accepted.

#### FREEZE

Set the freeze status indicator FLXAnFRCCSV.FSI to 1 and go to POC state HALT immediately. Can be called from any state.

#### SEND_MTS

Send single MTS symbol during the next following symbol window on the channel configured by FLXAnFRSUCC1.MTSA, FLXAnFRSUCC1.MTSB, when called in POC state NORMAL_ACTIVE after CC entered ALL slot mode (FLXAnFRCCSV.SLM = "11"). When called in any other state, or when called while a previously requested MTS has not yet been transmitted, FLXAnFRSUCC1.CMD will be reset to "$0000_B$" = command_not_accepted.

#### ALLOW_COLDSTART

The command resets FLXAnFRCCSV.CSI to enable the node to become leading coldstarter. When called in states DEFAULT_CONFIG, CONFIG, or HALT, FLXAnFRSUCC1.CMD will be reset to "$0000_B$" = command_not_accepted. To become leading coldstarter it is also required that both FLXAnFRSUCC1.TXST and FLXAnFRSUCC1.TXSY are set.

#### RESET_STATUS_INDICATORS

Resets status flags FLXAnFRCCSV.CSNI, FLXAnFRCCSV.CSAI, and FLXAnFRCCSV.WSV to the values after reset. May be called in POC states READY and STARTUP. When called in any other state, FLXAnFRSUCC1.CMD will be reset to "$0000_B$" = command_not_accepted.

#### CLEAR_RAMS

Sets FLXAnFRMHDS.CRAM to 1 when called in DEFAULT_CONFIG or CONFIG state. When called in any other state, FLXAnFRSUCC1.CMD will be reset to "$0000_B$" = command_not_accepted.

FLXAnFRMHDS.CRAM is also set to 1 when the CC leaves reset. By setting FLXAnFRMHDS.CRAM all internal RAM blocks are initialized to zero. During the initialization of the RAMs, FLXAnFRSUCC1.PBSY will show POC busy. Access to the configuration and status registers is possible during execution of CHI command CLEAR_RAMS.

The initialization of the internal message RAM requires 2048 bus clock cycles. There should be no Host access to IBF or OBF during initialization of the internal RAM blocks after reset or after assertion of CHI command CLEAR_RAMS.

Before asserting CHI command CLEAR_RAMS, the Host should make sure that no transfer between Message RAM and IBF / OBF or the Temporary Buffer RAMs is ongoing and that the data transfer handler has no effect (FLXAnFRITS.ITS = 0 and FLXAnFROTS.OTS = 0). This command also resets the Message Buffer Status registers FLXAnFRMHDS, FLXAnFRLDTS, FLXAnFRFSR, FLXAnFRMHDF, FLXAnFRTXRQ1/2/3/4, FLXAnFRNDAT1/2/3/4, and FLXAnFRMBSC1/2/3/4.

#### CAUTIONS

1. **All accepted commands with exception of CLEAR_RAMS and SEND_MTS will cause a change of the POC state in the FlexRay domain after at most 8 cycles of the slower of the two clocks "bus clock" and "FlexRay", assumed that POC was not busy when the command was applied and that no POC state change was forced by bus activity in that time frame. Reading register FLXAnFRCCSV will show data that is additionally delayed by synchronization from the FlexRay domain to the bus clock domain. The maximum additional delay is 12 cycles of the slower of the two clocks 'bus clock' and "FlexRay".**
2. **When the user restart as leading coldstarter after it has been stopped by FREEZE or READY command, it may happen, depending on the internal state of the FlexRay module, that the FlexRay module does not transmit its startup frame in cycle 0. Only FlexRay module configurations with startup frames configured for slots 1 to 7 are affected by this behaviour.**
   - **Coldstart after hardware reset is not affected.**
   - **Even if it happened, next coldstart attempt successful. Coldstart sequence is lengthened but coldstart of FlexRay system is not prohibited by this behaviour.**
   - **If the use wants to avoid this behavior, they must use a static slot greater or equal 8 for the startup / sync message.**

The below references the CHI commands from the FlexRay Protocol Specification to the FlexRay CHI command vector FLXAnFRSUCC1.CMD.

**Table 20.26 Reference to CHI Host Command Summary from FlexRay Protocol Specification**

| CHI command | Where processed (POC States) | CHI Command Vector CMD |
|---|---|---|
| ALL_SLOTS | POC: normal active, POC: normal passive | ALL_SLOTS |
| ALLOW_COLDSTART | All except POC: default config, POC: config, POC: halt | ALLOW_COLDSTART |
| CONFIG | POC: default config, POC: ready | CONFIG |
| CONFIG_COMPLETE | POC: config | Unlock sequence & READY |
| DEFAULT_CONFIG | POC: halt | CONFIG |
| FREEZE | All | FREEZE |
| HALT | POC: normal active, POC: normal passive | HALT |
| READY | All except POC: default config, POC: config, POC: ready, POC: halt | READY |
| RUN | POC: ready | RUN |
| WAKEUP | POC: ready | WAKEUP |

### 20.2.6.2 FLXAnFRSUCC2 — FlexRay SUC Configuration Register 2

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0084_H$

**Value after reset:** $0100\ 0504_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | LTN[3:0] | | | | — | — | — | LT[20:16] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.27 FLXAnFRSUCC2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 27 to 24 | LTN[3:0] | Listen Timeout Noise Bit<br>Configures (gListenNoise – 1) |
| 23 to 21 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 20 to 0 | LT[20:0] | Listen Timeout Bit<br>Configures pdListenTimeout |

### (1) FLXAnFRSUCC2.LTN

Listen Timeout Noise Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

The range for gListenNoise is 2 to 16.

FLXAnFRSUCC2.LTN must be configured identical in all nodes of a cluster.

Configures the upper limit for startup and wakeup listen timeout in the presence of noise expressed as a multiple of pdListenTimeout.

**CAUTION**

**The wakeup / startup noise timeout is calculated as follows:**

**pdListenTimeout × gListenNoise = FLXAnFRSUCC2.LT × (FLXAnFRSUCC2.LTN + 1)**

### (2) FLXAnFRSUCC2.LT

Listen Timeout Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

The range for pdListenTimeout is 1284 to 1283846 µT.

Configures wakeup / startup listen time out in µT.

### 20.2.6.3 FLXAnFRSUCC3 — FlexRay SUC Configuration Register 3

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0088_H$

**Value after reset:** $0000\ 0011_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | WCF[3:0] | | | | WCP[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.28 FLXAnFRSUCC3 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 7 to 4 | WCF[3:0] | Maximum Without Clock Correction Fatal Bit (transition to HALT state)<br>Configures gMaxWithoutClockCorrectionFatal |
| 3 to 0 | WCP[3:0] | Maximum Without Clock Correction Passive Bit (transition to NORMAL_PASSIVE state)<br>Configures gMaxWithoutClockCorrectionPassive |

**(1) FLXAnFRSUCC3.WCF**

Maximum Without Clock Correction Fatal Bit (transition to HALT state)

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 1 to 15 cycle pairs.

Must be identical in all nodes of a cluster.

Defines the number of consecutive even / odd cycle pairs with missing clock correction terms that will cause a transition from NORMAL_ACTIVE or NORMAL_PASSIVE to HALT state.

**CAUTION**

**The transition to HALT state is prevented if FLXAnFRSUCC1.HCSE is not set.**

**(2) FLXAnFRSUCC3.WCP**

Maximum Without Clock Correction Passive Bit (transition to NORMAL_PASSIVE state)

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 1 to 15 cycle pairs.

Must be identical in all nodes of a cluster.

Defines the number of consecutive even / odd cycle pairs with missing clock correction terms that will cause a transition from NORMAL_ACTIVE to NORMAL_PASSIVE state.

### 20.2.6.4 FLXAnFRNEMC — FlexRay NEM Configuration Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $008C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | NML[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 20.29 FLXAnFRNEMC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 3 to 0 | NML[3:0] | Network Management Vector Length Bit<br>Configures gNetworkManagementVectorLength |

#### (1) FLXAnFRNEMC.NML

Network Management Vector Length Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 12 bytes.

The configured length must be identical in all nodes of a cluster.

These bits configure the length of the NM vector.

### 20.2.6.5 FLXAnFRPRTC1 — FlexRay PRT Configuration Register 1

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0090_H$

**Value after reset:** $084C\ 0633_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RWP[5:0] | | | | | | — | RXW[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BRP[1:0] | | SPP[1:0] | | — | CASM[6:0] | | | | | | | TSST[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.30 FLXAnFRPRTC1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | RWP[5:0] | Repetitions of Tx Wakeup Pattern Bit<br>Configures pWakeupPattern |
| 25 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 24 to 16 | RXW[8:0] | Wakeup Symbol Receive Window Length Bit<br>Configures gdWakeupSymbolRxWindow |
| 15, 14 | BRP[1:0] | Baud Rate Prescaler Bit<br>Configures gdSampleClockPeriod and pSamplesPerMicrotick<br>00 = 10 Mbps<br>01 = 5 Mbps<br>10 = 2.5 Mbps<br>11 = 2.5 Mbps |
| 13, 12 | SPP[1:0] | Strobe Point Position Bit<br>Configures Strobe point position<br>00 = Sample 5<br>01 = Sample 4<br>10 = Sample 6<br>11 = Sample 5 |
| 11 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 10 to 4 | CASM[6:0] | Collision Avoidance Symbol Max Bit<br>Configures gdCASRxLowMax |
| 3 to 0 | TSST[3:0] | Transmission Start Sequence Transmitter Bit<br>Configures gdTSSTransmitter |

**(1) FLXAnFRPRTC1.RWP**

Repetitions of Tx Wakeup Pattern Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 63.

Configures the number of repetitions (sequences) of the Tx wakeup symbol.

**(2) FLXAnFRPRTC1.RXW**

Wakeup Symbol Receive Window Length Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 76 to 301 bit times.

Must be identical in all nodes of a cluster.

Configures the number of bit times used by the node to test the duration of the received wakeup pattern.

**(3) FLXAnFRPRTC1.BRP**

Baud Rate Prescaler Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

The Baud Rate Prescaler configures the baud rate on the FlexRay bus. The baud rates listed below are valid with a sample clock set to 80 MHz. One bit time always consists of 8 samples independent of the configured baud rate.

00 = 10 MBit/s

gdSampleClockPeriod = 12.5 ns = 1 * “sample clock”

pSamplesPerMicrotick = 2 (1 µT = 25 ns)

01 = 5 MBit/s

gdSampleClockPeriod = 25 ns = 2 * “sample clock”

pSamplesPerMicrotick = 1 (1 µT = 25 ns)

10, 11 = 2.5 MBit/s

gdSampleClockPeriod = 50 ns = 4 * “sample clock”

pSamplesPerMicrotick = 1 (1 µT = 50 ns)

### (4) FLXAnFRPRTC1.SPP

Strobe Point Position Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Defines the sample count value for strobing. The strobed bit value is set to the voted value when the sample count is incremented to the value configured by FLXAnFRPRTC1.SPP.

**CAUTION**

**The current revision 2.1 of the FlexRay protocol requires that FLXAnFRPRTC1.SPP = “00”. The alternate strobe point positions could be used to compensate for asymmetries in the physical layer.**

### (5) FLXAnFRPRTC1.CASM

Collision Avoidance Symbol Max Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

CASM6 is fixed to 1.

Valid values are 67 to 99 bit times.

Configures the upper limit of the acceptance window for a collision avoidance symbol (CAS).

### (6) FLXAnFRPRTC1.TSST

Transmission Start Sequence Transmitter Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 3 to 15 bit times.

Must be identical in all nodes of a cluster.

Configures the duration of the Transmission Start Sequence (TSS) in terms of bit times (1 bit time = 4 µT = 100ns @ 10Mbps).

### 20.2.6.6 FLXAnFRPRTC2 — FlexRay PRT Configuration Register 2

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0094_H$

**Value after reset:** $0F2D\ 0A0E_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TXL[5:0] | | | | | | TXI[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | RXL[5:0] | | | | | | — | — | RXI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.31 FLXAnFRPRTC2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 to 24 | TXL[5:0] | Wakeup Symbol Transmit Low Bit<br>Configures gdWakeupSymbolTxLow |
| 23 to 16 | TXI[7:0] | Wakeup Symbol Transmit Idle Bit<br>Configures gdWakeupSymbolTxIdle |
| 15, 14 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 13 to 8 | RXL[5:0] | Wakeup Symbol Receive Low Bit<br>Configures gdWakeupSymbolRxLow |
| 7, 6 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 5 to 0 | RXI[5:0] | Wakeup Symbol Rx Idle Bit<br>Configures gdWakeupSymbolRxIdle |

**(1) FLXAnFRPRTC2.TXL**

Wakeup Symbol Transmit Low Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 15 to 60 bit times.

Must be identical in all nodes of a cluster.

Configures the number of bit times used by the node to transmit the low phase of the wakeup symbol.

**(2) FLXAnFRPRTC2.TXI**

Wakeup Symbol Transmit Idle Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 45 to 180 bit times.

Must be identical in all nodes of a cluster.

Configures the number of bit times used by the node to transmit the idle phase of the wakeup symbol.

**(3) FLXAnFRPRTC2.RXL**

Wakeup Symbol Receive Low Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 10 to 55 bit times.

Must be identical in all nodes of a cluster.

Configures the number of bit times used by the node to test the duration of the low phase of the received wakeup symbol.

**(4) FLXAnFRPRTC2.RXI**

Wakeup Symbol Rx Idle Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 14 to 59 bit times.

Must be identical in all nodes of a cluster.

Configures the number of bit times used by the node to test the duration of the idle phase of the received wakeup symbol.

### 20.2.6.7 FLXAnFRMHDC — FlexRay MHD Configuration Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0098_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | SLT[12:0] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | SFDL[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.32 FLXAnFRMHDC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 28 to 16 | SLT[12:0] | Start of Latest Transmit Bit<br>Configures pLatestTx |
| 15 to 7 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 6 to 0 | SFDL[6:0] | Static Frame Data Length Bit<br>Configures gPayloadLengthStatic |

#### (1) FLXAnFRMHDC.SLT

Start of Latest Transmit Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 7981 minislots.

Configures the maximum minislot value allowed before inhibiting frame transmission in the dynamic segment of the cycle. There is no transmission in dynamic segment if FLXAnFRMHDC.SLT is set to zero.

#### (2) FLXAnFRMHDC.SFDL

Static Frame Data Length Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 127.

The payload length must be identical in all nodes of a cluster.

Configures the cluster-wide payload length for all frames sent in the static segment in double bytes.

### 20.2.6.8 FLXAnFRGTUC1 — FlexRay GTU Configuration Register 1

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $00A0_H$

**Value after reset:** $0000\ 0280_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | UT[19:16] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.33 FLXAnFRGTUC1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 19 to 0 | UT[19:0] | Setting of Communication Cycle in Microticks Bit<br>Configures pMicroPerCycle |

#### (1) FLXAnFRGTUC1.UT

Setting of Communication Cycle in Microticks Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 640 to 640000 µT.

Configures the duration of the communication cycle in microticks.

### 20.2.6.9 FLXAnFRGTUC2 — FlexRay GTU Configuration Register 2

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $00A4_H$

**Value after reset:** $0002\ 000A_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | SNM[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | MPC[13:0] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.34 FLXAnFRGTUC2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 19 to 16 | SNM[3:0] | Sync Node Max Bit |
| 15, 14 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 13 to 0 | MPC[13:0] | Setting of Communication Cycle in Macrotick Bit<br>Configures gMacroPerCycle |

#### (1) FLXAnFRGTUC2.SNM

Sync Node Max Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 15.

Must be identical in all nodes of a cluster.

Maximum number of frames within a cluster with sync frame indicator bit SYN set to 1.

#### (2) FLXAnFRGTUC2.MPC

Setting of Communication Cycle in Macrotick Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 10 to 16000 MT.

The cycle length must be identical in all nodes of a cluster.

Configures the duration of one communication cycle in macroticks.

### 20.2.6.10 FLXAnFRGTUC3 — FlexRay GTU Configuration Register 3

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $00A8_H$

**Value after reset:** $0202\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | MIOB[6:0] | | | | | | | — | MIOA[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UIOB[7:0] | | | | | | | | UIOA[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.35 FLXAnFRGTUC3 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 30 to 24 | MIOB[6:0] | Macrotick Initial Offset Channel B Bit<br>Configures pMacroInitialOffset[B] |
| 23 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 22 to 16 | MIOA[6:0] | Macrotick Initial Offset Channel A Bit<br>Configures pMacroInitialOffset[A] |
| 15 to 8 | UIOB[7:0] | Microtick Initial Offset Channel B Bit<br>Configures pMicroInitialOffset[B] |
| 7 to 0 | UIOA[7:0] | Microtick Initial Offset Channel A Bit<br>Configures pMicroInitialOffset[A] |

**(1) FLXAnFRGTUC3.MIOB**

Macrotick Initial Offset Channel B Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 72 MT.

Must be identical in all nodes of a cluster.

Configures the number of macroticks between the static slot boundary and the subsequent macrotick boundary of the secondary time reference point based on the nominal macrotick duration.

**(2) FLXAnFRGTUC3.MIOA**

Macrotick Initial Offset Channel A Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 72 MT.

Must be identical in all nodes of a cluster.

Configures the number of macroticks between the static slot boundary and the subsequent macrotick boundary of the secondary time reference point based on the nominal macrotick duration.

**(3) FLXAnFRGTUC3.UIOB**

Microtick Initial Offset Channel B Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 240 µT.

Configures the number of microticks between the actual time reference point on channel B and the subsequent macrotick boundary of the secondary time reference point. The parameter depends on pDelayCompensation [B] and therefore has to be set for each channel independently.

**(4) FLXAnFRGTUC3.UIOA**

Microtick Initial Offset Channel A Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 240 µT.

Configures the number of microticks between the actual time reference point on channel A and the subsequent macrotick boundary of the secondary time reference point. The parameter depends on pDelayCompensation [A] and therefore has to be set for each channel independently.

### 20.2.6.11 FLXAnFRGTUC4 — FlexRay GTU Configuration Register 4

For details about configuration of FLXAnFRGTUC4.NIT and FLXAnFRGTUC4.OCS see **Section 20.3.2.5, Configuration of NIT Start and Offset Correction Start**.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $00AC_H$

**Value after reset:** $0008\ 0007_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | OCS[13:0] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | NIT[13:0] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.36 FLXAnFRGTUC4 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 to 16 | OCS[13:0] | Offset Correction Start Bit<br>Configures (gOffsetCorrectionStart – 1) |
| 15, 14 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 13 to 0 | NIT[13:0] | Network Idle Time Start Bit<br>Configures (gMacroPerCycle -gdNIT – 1) |

**(1) FLXAnFRGTUC4.OCS**

Offset Correction Start Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 8 to 15998 MT.

For cluster consisting of E-Ray implementations only, it is sufficient to program FLXAnFRGTUC4.OCS = FLXAnFRGTUC4.NIT + 1.

Must be identical in all nodes of a cluster.

Determines the start of the offset correction within the NIT phase, calculated from start of cycle.

**(2) FLXAnFRGTUC4.NIT**

Network Idle Time Start Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 7 to 15997 MT.

Must be identical in all nodes of a cluster.

Configures the starting point of the Network Idle Time NIT at the end of the communication cycle expressed in terms of macroticks from the beginning of the cycle. The start of NIT is recognized if Macrotick = gMacroPerCycle-gdNIT -1 and the increment pulse of Macrotick is set.

### 20.2.6.12 FLXAnFRGTUC5 — FlexRay GTU Configuration Register 5

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $00B0_H$

**Value after reset:** $0E00\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEC[7:0] | | | | | | | | — | — | — | CDD[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCB[7:0] | | | | | | | | DCA[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.37 FLXAnFRGTUC5 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | DEC[7:0] | Decoding Correction Bit<br>Configures pDecodingCorrection |
| 23 to 21 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 20 to 16 | CDD[4:0] | Cluster Drift Damping Bit<br>Configures pClusterDriftDamping |
| 15 to 8 | DCB[7:0] | Delay Compensation Channel B Bit<br>Configures pDelayCompensation[B] |
| 7 to 0 | DCA[7:0] | Delay Compensation Channel A Bit<br>Configures pDelayCompensation[A] |

#### (1) FLXAnFRGTUC5.DEC

Decoding Correction Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 14 to 143 µT.

Configures the decoding correction value in microticks used to determine the primary time reference point.

#### (2) FLXAnFRGTUC5.CDD

Cluster Drift Damping Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 20 µT.

Configures the cluster drift damping value in microticks used in clock synchronization to minimize accumulation of rounding errors.

### (3) FLXAnFRGTUC5.DCB

Delay Compensation Channel B Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 200 µT.

Used to compensate for reception delays on channel B. This covers assumed propagation delay up to cPropagationDelayMax for microticks in the range of 0.0125 to 0.05µs. In practice, the minimum of the propagation delays of all sync nodes should be applied.

### (4) FLXAnFRGTUC5.DCA

Delay Compensation Channel A Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 200 µT.

Used to compensate for reception delays on channel A. This covers assumed propagation delay up to cPropagationDelayMax for microticks in the range of 0.0125 to 0.05µs. In practice, the minimum of the propagation delays of all sync nodes should be applied.

### 20.2.6.13 FLXAnFRGTUC6 — FlexRay GTU Configuration Register 6

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $00B4_H$

**Value after reset:** $0002\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | MOD[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | ASR[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.38 FLXAnFRGTUC6 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 26 to 16 | MOD[10:0] | Maximum Oscillator Drift Bit<br>Configures pdMaxDrift |
| 15 to 11 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 10 to 0 | ASR[10:0] | Accepted Startup Range Bit<br>Configures pdAcceptedStartupRange |

#### (1) FLXAnFRGTUC6.MOD

Maximum Oscillator Drift Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 1923 µT.

Maximum drift offset between two nodes that operate with unsynchronized clocks over one communication cycle in µT.

#### (2) FLXAnFRGTUC6.ASR

Accepted Startup Range Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 1875 µT.

Number of microticks constituting the expanded range of measured deviation for startup frames during integration.

### 20.2.6.14 FLXAnFRGTUC7 — FlexRay GTU Configuration Register 7

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $00B8_H$

**Value after reset:** $0002\ 0004_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | NSS[9:0] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | SSL[9:0] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.39 FLXAnFRGTUC7 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 25 to 16 | NSS[9:0] | Number of Static Slots Bit<br>Configures gNumberOfStaticSlots |
| 15 to 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 9 to 0 | SSL[9:0] | Static Slot Length Bit<br>Configures gdStaticSlot |

#### (1) FLXAnFRGTUC7.NSS

Number of Static Slots Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 1023.

The number of static slots must be identical in all nodes of a cluster.

Configures the number of static slots in a cycle.

#### (2) FLXAnFRGTUC7.SSL

Static Slot Length Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 4 to 659 MT.

The static slot length must be identical in all nodes of a cluster.

Configures the length of a static slot in macroticks.

### 20.2.6.15 FLXAnFRGTUC8 — FlexRay GTU Configuration Register 8

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $00BC_H$

**Value after reset:** $0000\ 0002_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | NMS[12:0] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | MSL[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.40 FLXAnFRGTUC8 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 28 to 16 | NMS[12:0] | Number of Minislots Bit<br>Configures gNumberOfMinislots |
| 15 to 6 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 5 to 0 | MSL[5:0] | Minislot Length Bit<br>Configures gdMinislot |

#### (1) FLXAnFRGTUC8.NMS

Number of Minislots Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 7986.

The number of minislots must be identical in all nodes of a cluster.

Configures the number of minislots within the dynamic segment of a cycle.

#### (2) FLXAnFRGTUC8.MSL

Minislot Length Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 63 MT.

The minislot length must be identical in all nodes of a cluster.

Configures the length of a minislot in macroticks.

### 20.2.6.16 FLXAnFRGTUC9 — FlexRay GTU Configuration Register 9

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $00C0_H$

**Value after reset:** $0000\ 0101_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DSI[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | MAPO[4:0] | | | | | — | — | APO[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.41 FLXAnFRGTUC9 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 17, 16 | DSI[1:0] | Dynamic Slot Idle Phase Bit<br>Configures gdDynamicSlotIdlePhase |
| 15 to 13 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 12 to 8 | MAPO[4:0] | Minislot Action Point Offset Bit<br>Configures gdMinislotActionPointOffset |
| 7, 6 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 5 to 0 | APO[5:0] | Action Point Offset Bit<br>Configures gdActionPointOffset |

**(1) FLXAnFRGTUC9.DSI**

Dynamic Slot Idle Phase Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 2 Minislot.

Must be identical in all nodes of a cluster.

Configures the duration of the dynamic slot idle phase in the number of minislots. The duration has to be greater or equal than the idle detection time.

**(2) FLXAnFRGTUC9.MAPO**

Minislot Action Point Offset Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 1 to 31 MT.

Must be identical in all nodes of a cluster.

Configures the action point offset in macroticks within the minislots of the dynamic segment.

**(3) FLXAnFRGTUC9.APO**

Action Point Offset Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 1 to 63 MT.

Must be identical in all nodes of a cluster.

Configures the action point offset in macroticks within static slots and symbol window.

### 20.2.6.17 FLXAnFRGTUC10 — FlexRay GTU Configuration Register 10

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $00C4_H$

**Value after reset:** $0002\ 0005_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | MRC[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | MOC[13:0] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.42 FLXAnFRGTUC10 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 26 to 16 | MRC[10:0] | Maximum Rate Correction Bit<br>Configures pRateCorrectionOut |
| 15, 14 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 13 to 0 | MOC[13:0] | Maximum Offset Correction Bit<br>Configures pOffsetCorrectionOut |

#### (1) FLXAnFRGTUC10.MRC

Maximum Rate Correction Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 1923 µT.

Holds the maximum permitted rate correction value to be applied by the internal clock synchronization algorithm. The CC checks only the internal rate correction value against the maximum rate correction value (absolute value).

#### (2) FLXAnFRGTUC10.MOC

Maximum Offset Correction Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 5 to 15266 µT.

Holds the maximum permitted offset correction value (absolute value) to be applied by the internal clock synchronization algorithm (absolute value). The CC checks only the internal offset correction value against the maximum offset correction value.

### 20.2.6.18 FLXAnFRGTUC11 — FlexRay GTU Configuration Register 11

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $00C8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | ERC[2:0] | | | — | — | — | — | — | EOC[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | ERCC[1:0] | | — | — | — | — | — | — | EOCC[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

**Table 20.43 FLXAnFRGTUC11 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 26 to 24 | ERC[2:0] | External Rate Correction Bit<br>Configures pExternRateCorrection |
| 23 to 19 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 18 to 16 | EOC[2:0] | External Offset Correction Bit<br>Configures pExternOffsetCorrection |
| 15 to 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 9, 8 | ERCC[1:0] | External Rate Correction Control Bit<br>Configures vExternRateControl<br>00: External rate correction is prohibited.<br>01: External rate correction is prohibited.<br>10: Subtract<br>11: Add |
| 7 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1, 0 | EOCC[1:0] | External Offset Correction Control Bit<br>Configures vExternOffsetControl<br>00: External offset correction is prohibited.<br>01: External offset correction is prohibited.<br>10: Subtract<br>11: Add |

**(1) FLXAnFRGTUC11.ERC**

External Rate Correction Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 7 µT.

Holds the external rate correction value in microticks to be applied by the internal clock synchronization algorithm. The value is subtracted / added from / to the calculated rate correction value. The value is applied during NIT.

**(2) FLXAnFRGTUC11.EOC**

External Offset Correction Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 7 µT.

Holds the external offset correction value in microticks to be applied by the internal clock synchronization algorithm. The value is subtracted / added from / to the calculated offset correction value. The value is applied during NIT.

**(3) FLXAnFRGTUC11.ERCC**

External Rate Correction Control Bit

Should be modified only outside NIT (Network Idle Time).

By writing to FLXAnFRGTUC11.ERCC the external rate correction is enabled as specified below.

00 = External rate correction is prohibited.

01 = External rate correction is prohibited.

10 = Subtract

External rate correction value subtracted from calculated rate correction value

11 = Add

External rate correction value added to calculated rate correction value

**(4) FLXAnFRGTUC11.EOCC**

External Offset Correction Control Bit

Should be modified only outside NIT (Network Idle Time).

By writing to FLXAnFRGTUC11.EOCC the external offset correction is enabled as specified below.

00 = External offset correction is prohibited.

01 = External offset correction is prohibited.

10 = Subtract

External offset correction value subtracted from calculated offset correction value

11 = Add

External offset correction value added to calculated offset correction value

## 20.2.7 CC Status Registers

During 8/16-bit accesses to status variables coded with more than 8/16-bit, the variable might be updated by the CC between two accesses (non-atomic read accesses).

### 20.2.7.1 FLXAnFRCCSV — FlexRay CC Status Vector Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0100_H$

**Value after reset:** $0010\ 4000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | PSL[5:0] | | | | | | RCA[4:0] | | | | | WSV[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CSI | CSAI | CSNI | — | — | SLM[1:0] | | HRQ | FSI | POCS[5:0] | | | | | |
| Value after reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.44 FLXAnFRCCSV Register Contents (1/2)**

| Bit | Symbol | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 to 24 | PSL[5:0] | POC Status Log Flag<br>Status of FLXAnFRCCSV.POCS immediately before entering HALT state. |
| 23 to 19 | RCA[4:0] | Remaining Coldstart Attempts Flag<br>Indicates vRemainingColdstartAttempts |
| 18 to 16 | WSV[2:0] | Wakeup Status Flag<br>Indicates vPOC!WakeupStatus<br>000: UNDEFINED<br>001: RECEIVED_HEADER<br>010: RECEIVED_WUP<br>011: COLLISION_HEADER<br>100: COLLISION_WUP<br>101: COLLISION_UNKNOWN<br>110: TRANSMITTED<br>111: Reserved |
| 15 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 14 | CSI | Cold Start Inhibit Flag<br>Indicates vColdStartInhibit<br>0: Cold starting of node enabled<br>1: Cold starting of node disabled |
| 13 | CSAI | Coldstart Abort Indicator Flag |
| 12 | CSNI | Coldstart Noise Indicator Flag<br>Indicates vPOC!ColdstartNoise |
| 11, 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |

Table 20.44 FLXAnFRCCSV Register Contents (2/2)

| Bit | Symbol | Function |
|---|---|---|
| 9, 8 | SLM[1:0] | Slot Mode Flag<br>Indicates vPOC!SlotMode<br>00: SINGLE<br>01: reserved<br>10: ALL_PENDING<br>11: ALL |
| 7 | HRQ | Halt Request Flag<br>Indicates vPOC!CHIHaltRequest |
| 6 | FSI | Freeze Status Indicator Flag<br>Indicates vPOC!Freeze |
| 5 to 0 | POCS[5:0] | Protocol Operation Control Status Flag |

### (1) FLXAnFRCCSV.PSL

POC Status Log Flag

Set the value of FLXAnFRCCSV.POCS immediately before entering HALT state.

Set to HALT when FREEZE command is applied during HALT state and FLXAnFRCCSV.FSI is not already set i.e. the HALT state was not reached by FREEZE command.

Reset to “B’000000” when leaving HALT state.

### (2) FLXAnFRCCSV.RCA

Remaining Coldstart Attempts Flag

Indicates the number of remaining coldstart attempts (vRemainingColdstartAttempts).

The value after a reset of FLXAnFRCCSV.RCA during CONFIG and DEFAULT_CONFIG state is also FLXAnFRSUCC1.CSA.

The RUN command resets this counter to the maximum number of coldstart attempts as configured by FLXAnFRSUCC1.CSA.

### (3) FLXAnFRCCSV.WSV

Wakeup Status Flag

Indicates the status of the current wakeup attempt (vPOC!WakeupStatus).

Reset to 0 when entering Wakeup state, by CHI command RESET_STATUS_INDICATORS, or by transition from DEFAULT_CONFIG to CONFIG state

$000_B$ = UNDEFINED

Wakeup not yet executed by the CC.

$001_B$ = RECEIVED_HEADER

Set when the CC finishes wakeup due to the reception of a frame header without coding violation on either channel in WAKEUP_LISTEN state.

$010_B$ = RECEIVED_WUP

Set when the CC finishes wakeup due to the reception of a valid wakeup pattern on the configured wakeup channel in WAKEUP_LISTEN state.

$011_B$ = COLLISION_HEADER

Set when the CC stops wakeup due to a detected collision during wakeup pattern transmission by receiving a valid header on either channel.

$100_B$ = COLLISION_WUP

Set when the CC stops wakeup due to a detected collision during wakeup pattern transmission by receiving a valid wakeup pattern on the configured wakeup channel.

$101_B$ = COLLISION_UNKNOWN

Set when the CC stops wakeup by leaving WAKEUP_DETECT state after expiration of the wakeup timer without receiving a valid wakeup pattern or a valid frame header.

$110_B$ = TRANSMITTED

Set when the CC has successfully completed the transmission of the wakeup pattern.

$111_B$ = reserved

**(4) FLXAnFRCCSV.CSI**

Cold Start Inhibit Flag

Indicates that the node is disabled from cold starting (vColdStartInhibit).

The flag is set to 1 whenever the POC enters READY state due to CHI command READY.

The flag has to be reset under control of the Host by CHI command ALLOW_COLDSTART (FLXAnFRSUCC1.CMD = “1001”).

**(5) FLXAnFRCCSV.CSAI**

Coldstart Abort Indicator Flag

Coldstart aborted.

Reset by CHI command RESET_STATUS_INDICATORS or by transition from HALT to DEFAULT_CONFIG state or from READY to STARTUP state.

**(6) FLXAnFRCCSV.CSNI**

Coldstart Noise Indicator Flag

Indicates that the cold start procedure occurred under noisy conditions (vPOC!ColdstartNoise).

Reset by CHI command RESET_STATUS_INDICATORS or by transition from HALT to DEFAULT_CONFIG state or from READY to STARTUP state.

**(7) FLXAnFRCCSV.SLM**

Slot Mode Flag

Indicates the actual slot mode of the POC (vPOC!SlotMode) in states READY, WAKEUP, STARTUP, NORMAL_ACTIVE, and NORMAL_PASSIVE.

Default value is SINGLE. Changes to ALL, depending on FLXAnFRSUCC1.TSM.

In NORMAL_ACTIVE or NORMAL_PASSIVE state the CHI command ALL_SLOTS will change the slot mode from SINGLE over ALL_PENDING to ALL.

Set FLXAnFRSUCC1.TSM to SINGLE except for NORMAL_ACTIVE or NORMAL_PASSIVE.

**(8) FLXAnFRCCSV.HRQ**

Halt Request Flag

Indicates that a request from the Host has been received to halt the POC at the end of the communication cycle (vPOC!CHIHaltRequest).

Reset by transition from HALT to DEFAULT_CONFIG state or when entering READY state.

**(9) FLXAnFRCCSV.FSI**

Freeze Status Indicator Flag

Indicates that the POC has entered the HALT state due to CHI command FREEZE or due to an error condition requiring an immediate POC halt (vPOC!Freeze).

Reset by transition from HALT to DEFAULT_CONFIG state.

**(10) FLXAnFRCCSV.POCS**

Protocol Operation Control Status Flag

Indicates the actual state of operation of the CC Protocol Operation Control

$00\ 0000_B$ = DEFAULT_CONFIG state

$00\ 0001_B$ = READY state

$00\ 0010_B$ = NORMAL_ACTIVE state

$00\ 0011_B$ = NORMAL_PASSIVE state

$00\ 0100_B$ = HALT state

$00\ 0110_B$…$00\ 1110_B$ = reserved

$00\ 1111_B$ = CONFIG state

Indicates the actual state of operation of the POC in the wakeup path

$01\ 0000_B$ = WAKEUP_STANDBY state

$01\ 0001_B$ = WAKEUP_LISTEN state

$01\ 0010_B$ = WAKEUP_SEND state

$01\ 0011_B$ = WAKEUP_DETECT state

Indicates the actual state of operation of the POC in the startup path

$10\ 0000_B$ = STARTUP_PREPARE state

$10\ 0001_B$ = COLDSTART_LISTEN state

$10\ 0010_B$ = COLDSTART_COLLISION_RESOLUTION state

$10\ 0011_B$ = COLDSTART_CONSISTENCY_CHECK state

$10\ 0100_B$ = COLDSTART_GAP state

$10\ 0101_B$ = COLDSTART_JOIN State

$10\ 0110_B$ = INTEGRATION_COLDSTART_CHECK state

$10\ 0111_B$ = INTEGRATION_LISTEN state

$10\ 1000_B$ = INTEGRATION_CONSISTENCY_CHECK state

$10\ 1001_B$ = INITIALIZE_SCHEDULE state

$10\ 1010_B$ = ABORT_STARTUP state

$10\ 1011_B$ = STARTUP_SUCCESS state

others = reserved

### 20.2.7.2 FLXAnFRCCEV — FlexRay CC Error Vector Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0104_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | PTAC[4:0] | | | | | ERRM[1:0] | | — | — | CCFC[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.45 FLXAnFRCCEV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 12 to 8 | PTAC[4:0] | Passive to Active Count Flag<br>Indicates vAllowPassiveToActive |
| 7, 6 | ERRM[1:0] | Error Mode Flag<br>Indicates vPOC!ErrorMode<br>00: ACTIVE<br>01: PASSIVE<br>10: COMM_HALT<br>11: reserved |
| 5, 4 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 3 to 0 | CCFC[3:0] | Clock Correction Failed Counter<br>Indicates vClockCorrectionFailed |

**(1) FLXAnFRCCEV.PTAC**

Passive to Active Count Flag

Indicates the number of consecutive even / odd cycle pairs that have passed with valid rate and offset correction terms, while the node is waiting to transit from NORMAL_PASSIVE state to NORMAL_ACTIVE state. The transition takes place when FLXAnFRCCEV.PTAC equals FLXAnFRSUCC1.PTA –1.

Reset by transition from HALT to DEFAULT_CONFIG state or when entering READY state.

**(2) FLXAnFRCCEV.ERRM**

Error Mode Flag

Indicates the actual error mode of the POC (vPOC!ErrorMode).

Reset by transition from HALT to DEFAULT_CONFIG state or when entering READY state.

**(3) FLXAnFRCCEV.CCFC**

Clock Correction Failed Counter

Indicates the clockcorrection failed counter of the POC (vClockCorrectionFailed).

The Clock Correction Failed Counter is incremented by one at the end of any odd communication cycle where either the missing offset correction error or missing rate correction error are active.

The Clock Correction Failed Counter is reset to 0 at the end of an odd communication cycle if neither the offset correction failed nor the rate correction failed errors are active.

The Clock Correction Failed Counter stops at 15.

Reset by transition from HALT to DEFAULT_CONFIG state or when entering READY state.

### 20.2.7.3 FLXAnFRSCV — FlexRay Slot Counter Value Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0110_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SCCB[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SCCA[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.46 FLXAnFRSCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 26 to 16 | SCCB[10:0] | Slot Counter Channel B<br>Indicates vSlotCounter[B] |
| 15 to 11 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 10 to 0 | SCCA[10:0] | Slot Counter Channel A<br>Indicates vSlotCounter[A] |

#### (1) FLXAnFRSCV.SCCB

Slot Counter Channel B

Current slot counter value on channel B (vSlotCounter[B]). The value is incremented by the CC and reset at the start of a communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

#### (2) FLXAnFRSCV.SCCA

Slot Counter Channel A

Current slot counter value on channel A (vSlotCounter[A]). The value is incremented by the CC and reset at the start of a communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

### 20.2.7.4 FLXAnFRMTCCV — FlexRay Macrotick and Cycle Counter Value Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0114_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CCV[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | MTV[13:0] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.47 FLXAnFRMTCCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 21 to 16 | CCV[5:0] | Cycle Counter Value<br>Indicates vCycleCounter |
| 15, 14 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 13 to 0 | MTV[13:0] | Macrotick Value<br>Indicates vMacrotick |

#### (1) FLXAnFRMTCCV.CCV

Cycle Counter Value

Current cycle counter value (vCycleCounter). The value is incremented by the CC at the start of a communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

#### (2) FLXAnFRMTCCV.MTV

Macrotick Value

Current macrotick value (vMacrotick). The value is incremented by the CC and reset at the start of a communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

### 20.2.7.5 FLXAnFRRCV — FlexRay Rate Correction Value Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0118_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | RCV[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.48 FLXAnFRRCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 12 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 11 to 0 | RCV[11:0] | Rate Correction Value Flag<br>Indicates vRateCorrection |

#### (1) FLXAnFRRCV.RCV

Rate Correction Value Flag

Indicates internal rate correction value (vRateCorrection/ two's complement) before limitation. If the FLXAnFRRCV.RCV value exceeds the limits defined by FLXAnFRGTUC10.MRC, flag FLXAnFRSFS.RCLR is set to 1.

**CAUTION**

The external rate correction value is added to the limited rate correction value.

### 20.2.7.6 FLXAnFROCV — FlexRay Offset Correction Value Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $011C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | OCV[18:16] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OCV[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.49 FLXAnFROCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 18 to 0 | OCV[18:0] | Offset Correction Value Flag<br>Indicates vOffsetCorrection |

#### (1) FLXAnFROCV.OCV

Offset Correction Value Flag

Indicates offset correction value (vOffsetCorrection/two's complement) before limitation. If the FLXAnFROCV.OCV value exceeds the limits defined by FLXAnFRGTUC10.MOC, flag FLXAnFRSFS.OCLR is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

**CAUTION**

The external offset correction value is added to the limited offset correction value.

### 20.2.7.7 FLXAnFRSFS — FlexRay Sync Frame Status Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0120_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | RCLR | MRCS | OCLR | MOCS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VSBO[3:0] | | | | VSBE[3:0] | | | | VSAO[3:0] | | | | VSAE[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.50 FLXAnFRSFS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 19 | RCLR | Rate Correction Limit Reached Flag<br>0: Rate correction below limit<br>1: Rate correction limit reached |
| 18 | MRCS | Missing Rate Correction Signal Flag<br>0: Rate correction signal valid<br>1: Missing rate correction signal |
| 17 | OCLR | Offset Correction Limit Reached Flag<br>0: Offset correction below limit<br>1: Offset correction limit reached |
| 16 | MOCS | Missing Offset Correction Signal Flag<br>0: Offset correction signal valid<br>1: Missing offset correction signal |
| 15 to 12 | VSBO[3:0] | Valid Sync Frames Channel B, odd communication cycle |
| 11 to 8 | VSBE[3:0] | Valid Sync Frames Channel B, even communication cycle |
| 7 to 4 | VSAO[3:0] | Valid Sync Frames Channel A, odd communication cycle |
| 3 to 0 | VSAE[3:0] | Valid Sync Frames Channel A, even communication cycle |

**(1) FLXAnFRSFS.RCLR**

Rate Correction Limit Reached Flag

The Rate Correction Limit Reached flag signals to the Host, that the rate correction value has exceeded its limit as defined by FLXAnFRGTUC10.MRC10 to MRC0. The flag is updated by the CC at start of offset correction phase.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

**(2) FLXAnFRSFS.MRCS**

Missing Rate Correction Signal Flag

The Missing Rate Correction flag signals to the Host, that no rate correction calculation can be performed because no pairs of even / odd sync frames were received. The flag is updated by the CC at start of offset correction phase.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

**(3) FLXAnFRSFS.OCLR**

Offset Correction Limit Reached Flag

The Offset Correction Limit Reached flag signals to the Host, that the offset correction value has exceeded its limit as defined by FLXAnFRGTUC10.MOC. The flag is updated by the CC at start of offset correction phase.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

**(4) FLXAnFRSFS.MOCS**

Missing Offset Correction Signal Flag

The Missing Offset Correction flag signals to the Host, that no offset correction calculation can be performed because no sync frames were received. The flag is updated by the CC at start of offset correction phase.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

**(5) FLXAnFRSFS.VSBO**

Valid Sync Frames Channel B, odd communication cycle

These bits are only valid when FLXAnFRSUCC1.CCHB is 1.

Holds the number of valid sync frames received on channel B in the odd communication cycle. If transmission of sync frames is enabled by FLXAnFRSUCC1.TXSY the value is incremented by one. The value is updated during the NIT of each odd communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

**(6) FLXAnFRSFS.VSBE**

Valid Sync Frames Channel B, even communication cycle

These bits are only valid when FLXAnFRSUCC1.CCHB is 1.

Holds the number of valid sync frames received on channel B in the even communication cycle. If transmission of sync frames is enabled by FLXAnFRSUCC1.TXSY the value is incremented by one. The value is updated during the NIT of each even communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

**(7) FLXAnFRSFS.VSAO**

Valid Sync Frames Channel A, odd communication cycle

These bits are only valid when FLXAnFRSUCC1.CCHA is 1.

Holds the number of valid sync frames received on channel A in the odd communication cycle. If transmission of sync frames is enabled by FLXAnFRSUCC1.TXSY the value is incremented by one. The value is updated during the NIT of each odd communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

**(8) FLXAnFRSFS.VSAE**

Valid Sync Frames Channel A, even communication cycle

These bits are only valid when FLXAnFRSUCC1.CCHA is 1.

Holds the number of valid sync frames received on channel A in the even communication cycle. If transmission of sync frames is enabled by FLXAnFRSUCC1.TXSY the value is incremented by one. The value is updated during the NIT of each even communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

### 20.2.7.8 FLXAnFRSWNIT — FlexRay Symbol Window and NIT Status Register

Symbol window related status information is updated by the CC at the end of the symbol window for each channel. NIT related status information is updated by the CC at the end of the NIT for each channel.

During startup the status data is not updated.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0124_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SBNB | SENB | SBNA | SENA | MTSB | MTSA | TCSB | SBSB | SESB | TCSA | SBSA | SESA |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.51 FLXAnFRSWNIT Register Contents (1/2)**

| Bit | Symbol | Function |
|---|---|---|
| 31 to 12 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 11 | SBNB | Slot Boundary Violation during NIT Channel B Flag<br>0: No slot boundary violation detected<br>1: Slot boundary violation during NIT detected on channel B |
| 10 | SENB | Syntax Error during NIT Channel B Flag<br>0: No syntax error detected<br>1: Syntax error during NIT detected on channel B |
| 9 | SBNA | Slot Boundary Violation during NIT Channel A Flag<br>0: No slot boundary violation detected<br>1: Slot boundary violation during NIT detected on channel A |
| 8 | SENA | Syntax Error during NIT Channel A Flag<br>0: No syntax error detected<br>1: Syntax error during NIT detected on channel A |
| 7 | MTSB | MTS Received on Channel B Flag<br>0: No MTS symbol received on channel B<br>1: MTS symbol received on channel B |
| 6 | MTSA | MTS Received on Channel A Flag<br>0: No MTS symbol received on channel A<br>1: MTS symbol received on channel A |
| 5 | TCSB | Transmission Conflict in Symbol Window Channel B Flag<br>0: No transmission conflict detected<br>1: Transmission conflict in symbol window detected on channel B |
| 4 | SBSB | Slot Boundary Violation in Symbol Window Channel B Flag<br>0: No slot boundary violation detected<br>1: Slot boundary violation during symbol window detected on channel B |
| 3 | SESB | Syntax Error in Symbol Window Channel B Flag<br>0: No syntax error detected<br>1: Syntax error during symbol window detected on channel B |

Table 20.51 FLXAnFRSWNIT Register Contents (2/2)

| Bit | Symbol | Function |
|---|---|---|
| 2 | TCSA | Transmission Conflict in Symbol Window Channel A Flag<br>0: No transmission conflict detected<br>1: Transmission conflict in symbol window detected on channel A |
| 1 | SBSA | Slot Boundary Violation in Symbol Window Channel A Flag<br>0: No slot boundary violation detected<br>1: Slot boundary violation during symbol window detected on channel A |
| 0 | SESA | Syntax Error in Symbol Window Channel A Flag<br>0: No syntax error detected<br>1: Syntax error during symbol window detected on channel A |

**(1) FLXAnFRSWNIT.SBNB**

Indicates a Slot Boundary Violation during NIT Channel B Flag (vSS!BViolationB).

Reset when leaving CONFIG state or when entering STARTUP state.

**(2) FLXAnFRSWNIT.SENB**

Indicates a Syntax Error during NIT Channel B Flag (vSS!SyntaxErrorB).

Reset when leaving CONFIG state or when entering STARTUP state.

**(3) FLXAnFRSWNIT.SBNA**

Indicates a Slot Boundary Violation during NIT Channel A Flag (vSS!BViolationA).

Reset when leaving CONFIG state or when entering STARTUP state.

**(4) FLXAnFRSWNIT.SENA**

Indicates a Syntax Error during NIT Channel A Flag (vSS!SyntaxErrorA).

Reset when leaving CONFIG state or when entering STARTUP state.

**(5) FLXAnFRSWNIT.MTSB**

Indicates a MTS Received on Channel B Flag (vSS!ValidMTSB).

Media Access Test symbol received on channel B during the preceding symbol window. Updated by the CC for each channel at the end of the symbol window.

When this bit is set to 1, also interrupt flag FLXAnFRSIR.MTSB is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

**(6) FLXAnFRSWNIT.MTSA**

Indicates a MTS Received on Channel A Flag (vSS!ValidMTSA).

Media Access Test symbol received on channel A during the preceding symbol window. Updated by the CC for each channel at the end of the symbol window.

When this bit is set to 1, also interrupt flag FLXAnFRSIR.MTSA is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

**(7) FLXAnFRSWNIT.TCSB**

Indicates a Transmission Conflict in Symbol Window Channel B Flag (vSS!TxConflictB).

Reset when leaving CONFIG state or when entering STARTUP state.

**(8) FLXAnFRSWNIT.SBSB**

Indicates a Slot Boundary Violation in Symbol Window Channel B Flag (vSS!BViolationB).

Reset when leaving CONFIG state or when entering STARTUP state.

**(9) FLXAnFRSWNIT.SESB**

Indicates a Syntax Error in Symbol Window Channel B Flag (vSS!SyntaxErrorB).

Reset when leaving CONFIG state or when entering STARTUP state.

**(10) FLXAnFRSWNIT.TCSA**

Indicates a Transmission Conflict in Symbol Window Channel A Flag (vSS!TxConflictA).

Reset when leaving CONFIG state or when entering STARTUP state.

**(11) FLXAnFRSWNIT.SBSA**

Indicates a Slot Boundary Violation in Symbol Window Channel A Flag (vSS!BViolationA).

Reset when leaving CONFIG state or when entering STARTUP state.

**(12) FLXAnFRSWNIT.SESA**

Indicates a Syntax Error in Symbol Window Channel A Flag (vSS!SyntaxErrorA).

Reset when leaving CONFIG state or when entering STARTUP state.

### 20.2.7.9 FLXAnFRACS — FlexRay Aggregated Channel Status Register

The aggregated channel status provides the Host with an accrued status of channel activity for all communication slots regardless of whether they are assigned for transmission or subscribed for reception.

The aggregated channel status also includes status data from the symbol window and the network idle time.

The status data is updated (set) after each slot and aggregated until it is reset by the Host.

During startup the status data is not updated.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0128_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | SBVB | CIB | CEDB | SEDB | VFRB | — | — | — | SBVA | CIA | CEDA | SEDA | VFRA |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 20.52 FLXAnFRACS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 12 | SBVB | Slot Boundary Violation on Channel B Flag<br>0: No slot boundary violation observed<br>1: Slot boundary violation(s) observed on channel B |
| 11 | CIB | Communication Indicator Channel B Flag<br>0: No valid frame(s) received in slots containing any additional communication<br>1: Valid frame(s) received on channel B in slots containing any additional communication |
| 10 | CEDB | Content Error Detected on Channel B Flag<br>0: No frame with content error received<br>1: Frame(s) with content error received on channel B |
| 9 | SEDB | Syntax Error Detected on Channel B Flag<br>0: No syntax error observed<br>1: Syntax error(s) observed on channel B |
| 8 | VFRB | Valid Frame Received on Channel B Flag<br>0: No valid frame received<br>1: Valid frame(s) received on channel B |
| 7 to 5 | Reserved | These bits are always read as 0. The write value should be always 0. |
| 4 | SBVA | Slot Boundary Violation on Channel A Flag<br>0: No slot boundary violation observed<br>1: Slot boundary violation(s) observed on channel A |
| 3 | CIA | Communication Indicator Channel A Flag<br>0: No valid frame(s) received in slots containing any additional communication<br>1: Valid frame(s) received on channel A in slots containing any additional communication |

Table 20.52 FLXAnFRACS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | CEDA | Content Error Detected on Channel A Flag<br>0: No frame with content error received<br>1: Frame(s) with content error received on channel A |
| 1 | SEDA | Syntax Error Detected on Channel A Flag<br>0: No syntax error observed<br>1: Syntax error(s) observed on channel A |
| 0 | VFRA | Valid Frame Received on Channel A Flag<br>0: No valid frame received<br>1: Valid frame(s) received on channel A |

### (1) FLXAnFRACS.SBVB

Slot Boundary Violation on Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

One or more slot boundary violations were observed on channel B at any time during the observation period (static or dynamic slots, symbol window, and NIT).

When this flag changes from 0 to 1, interrupt flag FLXAnFREIR.EDB is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

### (2) FLXAnFRACS.CIB

Communication Indicator Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

One or more valid frames were received on channel B in slots that also contained any additional communication during the observation period, i.e. one or more slots received a valid frame AND had any combination of either syntax error OR content error OR slot boundary violation.

**CAUTION**

The set condition of the flag FLXAnFRACS.CIB is also fulfilled if there is only one single frame in the slot and the slot boundary at the end of the slot is reached during the frames channel idle recognition phase.
When this flag changes from 0 to 1, interrupt flag FLXAnFREIR.EDB is set to 1.
Reset when leaving CONFIG state or when entering STARTUP state.

### (3) FLXAnFRACS.CEDB

Content Error Detected on Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

One or more frames with a content error were received on channel B in any static or dynamic slot during the observation period.

When this flag changes from 0 to 1, interrupt flag FLXAnFREIR.EDB is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

#### (4) FLXAnFRACS.SEDB

Syntax Error Detected on Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

One or more syntax errors in static or dynamic slots, symbol window, and NIT were observed on channel B.

When this flag changes from 0 to 1, interrupt flag FLXAnFREIR.EDB is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

#### (5) FLXAnFRACS.VFRB

Valid Frame Received on Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

One or more valid frames were received on channel B in any static or dynamic slot during the observation period.

Reset when leaving CONFIG state or when entering STARTUP state.

#### (6) FLXAnFRACS.SBVA

Slot Boundary Violation on Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

Slot boundary violations were observed on channel A at any time during the observation period (static or dynamic slots, symbol window, and NIT).

When this flag changes from 0 to 1, interrupt flag FLXAnFREIR.EDA is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

#### (7) FLXAnFRACS.CIA

Communication Indicator Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

One or more valid frames were received on channel A in slots that also contained any additional communication during the observation period, i.e. one or more slots received a valid frame AND had any combination of either syntax error OR content error OR slot boundary violation.

**CAUTION**

The set condition of the flag FLXAnFRACS.CIA is also fulfilled if there is only one single frame in the slot and the slot boundary at the end of the slot is reached during the frames channel idle recognition phase.

When this flag changes from 0 to 1, interrupt flag FLXAnFREIR.EDA is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

**(8) FLXAnFRACS.CEDA**

Content Error Detected on Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

One or more frames with a content error were received on channel A in any static or dynamic slot during the observation period.

When this flag changes from 0 to 1, interrupt flag FLXAnFREIR.EDA is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

**(9) FLXAnFRACS.SEDA**

Syntax Error Detected on Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

One or more syntax errors in static or dynamic slots, symbol window, and NIT were observed on channel A.

When this flag changes from 0 to 1, interrupt flag FLXAnFREIR.EDA is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

**(10) FLXAnFRACS.VFRA**

Valid Frame Received on Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

One or more valid frames were received on channel A in any static or dynamic slot during the observation period.

Reset when leaving CONFIG state or when entering STARTUP state.

### 20.2.7.10 FLXAnFRESIDm — FlexRay Even Sync ID Register m (m = 1 to 15)

Registers FLXAnFRESID1 to FLXAnFRESID15 hold the frame IDs of the sync frames received in even communication cycles used for clock synchronisation up to the limit of gSyncNodeMax. The values are sorted in ascending order, with register FLXAnFRESID1 holding the lowest received sync frame ID. If the node itself transmits a sync frame in an even communication cycle, register FLXAnFRESID1 holds the respective sync frame ID as configured in message buffer 0 and flags FLXAnFRESID1.RXEA, FLXAnFRESID1.RXEB are set. The value is updated during the NIT of each even communication cycle.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0130_H$ to <FLXn_base> + $0168_H$ (<FLXn_base> + $0130_H$ + (n – 1) *4)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RXEB | RXEA | — | — | — | — | EID[9:0] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.53 FLXAnFRESIDn (n = 1 to 15) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 15 | RXEB | Received / Configured Even Sync ID on Channel B Flag<br>0: No sync frame received on channel B / node not configured to transmit sync frames<br>1: Sync frame received on channel B / node configured to transmit sync frames |
| 14 | RXEA | Received / Configured Even Sync ID on Channel A Flag<br>0: No sync frame received on channel A / node not configured to transmit sync frames<br>1: Sync frame received on channel A / node configured to transmit sync frames |
| 13 to 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 9 to 0 | EID[9:0] | Even Sync ID (vsSyncIDListA, B even) Flag |

**(1) FLXAnFRESIDn.RXEB**

Received / Configured Even Sync ID on Channel B Flag

Signals that a sync frame corresponding to the stored even sync ID was received on channel B or that the node is configured to be a sync node with key slot = FLXAnFRESID1.EID.

Reset when leaving CONFIG state or when entering STARTUP state.

**(2) FLXAnFRESIDn.RXEA**

Received / Configured Even Sync ID on Channel A Flag

Signals that a sync frame corresponding to the stored even sync ID was received on channel A or that the node is configured to be a sync node with key slot = FLXAnFRESID1.EID.

Reset when leaving CONFIG state or when entering STARTUP state.

**(3) FLXAnFRESIDn.EID**

Even Sync ID Flag (vsSyncIDListA, B even)

Sync frame ID even communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

### 20.2.7.11 FLXAnFROSIDm — FlexRay Odd Sync ID Register m (m = 1 to 15)

Registers FLXAnFROSID1 to FLXAnFROSID15 hold the frame IDs of the sync frames received in odd communication cycles used for clock synchronisation up to the limit of gSyncNodeMax. The values are sorted in ascending order, with register FLXAnFROSID1 holding the lowest received sync frame ID. If the node itself transmits a sync frame in an odd communication cycle, register FLXAnFROSID1 holds the respective sync frame ID as configured in message buffer 0 and flags FLXAnFROSID1.RXOA, FLXAnFROSID1.RXOB are set. The value is updated during the NIT of each odd communication cycle.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0170_H$ to <FLXn_base> + $01A8_H$ (<FLXn_base> + $0170_H$ + (n – 1)*4)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RXOB | RXOA | — | — | — | — | OID[9:0] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.54 FLXAnFROSIDn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 15 | RXOB | Received / Configured Odd Sync ID on Channel B Flag<br>0: No sync frame received on channel B / node not configured to transmit sync frames<br>1: Sync frame received on channel B / node configured to transmit sync frames |
| 14 | RXOA | Received / Configured Odd Sync ID on Channel A Flag<br>0: No sync frame received on channel A / node not configured to transmit sync frames<br>1: Sync frame received on channel A / node configured to transmit sync frames |
| 13 to 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 9 to 0 | OID[9:0] | Odd Sync ID Flag (vsSyncIDListA, B odd) |

**(1) FLXAnFROSIDn.RXOB**

Received / Configured Odd Sync ID on Channel B Flag

Signals that a sync frame corresponding to the stored odd sync ID was received on channel B or that the node is configured to be a sync node with key slot = FLXAnFROSID1.OID.

Reset when leaving CONFIG state or when entering STARTUP state.

**(2) FLXAnFROSIDn.RXOA**

Received / Configured Odd Sync ID on Channel A Flag

Signals that a sync frame corresponding to the stored odd sync ID was received on channel A or that the node is configured to be a sync node with key slot = FLXAnFROSID1.OID.

Reset when leaving CONFIG state or when entering STARTUP state.

**(3) FLXAnFROSIDn.OID**

Odd Sync ID Flag (vsSyncIDListA, B odd)

Sync frame ID odd communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

### 20.2.7.12 FLXAnFRNMVm — FlexRay Network Management Vector Register m (m = 1 to 3)

The three network management registers hold the accrued NM vector (see **Section 20.3.7, Network Management**).

The CC updates the NM vector at the end of each communication cycle as long as the CC is either in NORMAL_ACTIVE or NORMAL_PASSIVE state.

NMVn-bytes exceeding the configured NM vector length are not valid.

For information about the byte alignment of the received NM vector in this register see **Section 20.3.17, Byte Alignment**.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $01B0_H$ to <FLXn_base> + $01B8_H$ (<FLXn_base> + $01B0_H$ + (n – 1)*4)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NM[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.55 FLXAnFRNMVn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | NM[31:0] | NM Vector<br>The three network management vector registers hold the accrued NM vector (configurable 0 to 12 bytes). The NM vector to be held is generated by bit-wise logic OR for each NM vector received on each channel (valid static frames with PPI = 1) (see **Section 20.3.7, Network Management**).<br>For information about the byte alignment of the received NM vector in this register see **Section 20.3.17, Byte Alignment**.<br>NMVn-bytes exceeding the configured NM vector length are not valid.<br>The Register Contents are updated at the end of each communication cycle as long as the CC is either in NORMAL_ACTIVE or NORMAL_PASSIVE state.<br>These bits are cleared when leaving CONFIG state or when entering STARTUP state. |

## 20.2.8 Message Buffer Control Registers

### 20.2.8.1 FLXAnFRMRC — FlexRay Message RAM Configuration Register

The Message RAM Configuration register defines the number of message buffers assigned to the static segment, dynamic segment, and FIFO.

The Message RAM can be divided into up three different areas; Static Buffer area, Static and Dynamic Buffer area, FIFO area. If present, the Static Buffer area is starting at Message Buffer 0.

The start of the Static and Dynamic Buffer area is configured by FLXAnFRMRC.FDB. FLXAnFRMRC.FDB defines the end of the Static Buffer area. If no Static Buffer area is present, the Static and Dynamic Buffer area starts at Message Buffer 0.

The start of the FIFO area is configured by FLXAnFRMRC.FFB. FLXAnFRMRC.FFB defines the end of the previous area, which can be either the Static Buffer area or the Static and Dynamic Buffer area. If no Static Buffer area and no Static and Dynamic Buffer area is present, the FIFO area starts at Message Buffer 0.

With FLXAnFRMRC.LCB the end of the last configured area is configured which can be the Static Buffer area, the Static and Dynamic Buffer area or the FIFO area.

**Figure 20.2** shows an example configuration of the Message RAM where all there area are configured.

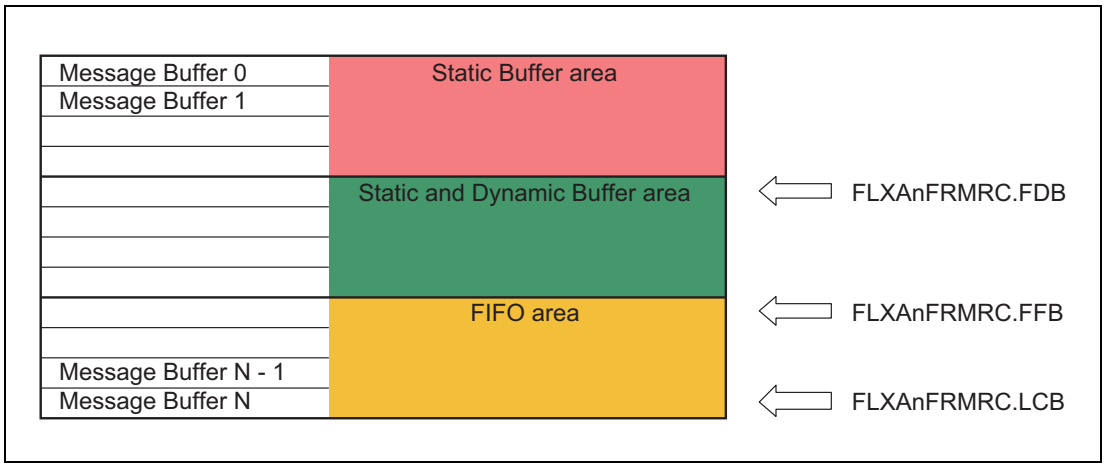



Figure 20.2 FLXAnFRNMVn Register Contents

**CAUTIONS**

1. **In case the node is configured as sync node (FLXAnFRSUCC1.TXSY = 1) or for single slot mode operation (FLXAnFRSUCC1.TSM = 1), message buffer 0 resp. 1 is reserved for sync frames or single slot frames and have to be configured with the node-specific key slot ID. In case the node is neither configured as sync node nor for single slot operation message buffer 0 resp. 1 is treated like all other message buffers.**
2. **The maximum number of header sections is 128. This means a maximum of 128 message buffers can be configured. The maximum length of a data section is 254 bytes. The length of the data section may be configured differently for each message buffer. For details see Section 20.3.13, Message RAM.**
3. **In case two or more message buffers are assigned to slot 1 by use of cycle filtering, all of them must be located either in the “Static Buffers” or at the**

**beginning of the “Static + Dynamic Buffers” section.**

4. **The FlexRay protocol specification requires that each node has to send a frame in its key slot. Therefore at least message buffer 0 is reserved for transmission in the key slot. Due to this requirement a maximum number of 127 message buffers can be assigned to the FIFO. Nevertheless, a non protocol conform configuration without a transmission slot in the static segment would still be operational.**
5. **The payload length configured and the length of the data section need to be configured identical for all message buffers belonging to the FIFO via FLXAnFRWRHS2.PL and FLXAnFRWRHS3.DP. When the CC is not in DEFAULT_CONFIG or CONFIG state reconfiguration of message buffers belonging to the FIFO is locked.**

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0300_H$

**Value after reset:** $0180\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SPLM | SEC[1:0] | | LCB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FFB[7:0] | | | | | | | | FDB[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.56 FLXAnFRMRC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 26 | SPLM | Sync Frame Payload Multiplex Bit<br>0: Only message buffer 0 locked against reconfiguration<br>1: Both message buffers 0 and 1 are locked against reconfiguration |
| 25, 24 | SEC[1:0] | Secure Buffers Bit<br>00: all buffers unlocked<br>01: static buffers locked, FIFO locked, limited transmission<br>10: all buffers locked<br>11: all buffers locked, limited transmission |
| 23 to 16 | LCB[7:0] | Last Configured Buffer Bit<br>0 to 127: Number of message buffers is FLXAnFRMRF.LCB + 1<br>128: No message buffer assigned to the FIFO |
| 15 to 8 | FFB[7:0] | First Buffer of FIFO Bit<br>0: All message buffers assigned to the FIFO<br>1 to 127: Message buffers from FLXAnFRMRC.FFB to FLXAnFRMRC.LCB assigned to the FIFO<br>128: No message buffer configured |
| 7 to 0 | FDB[7:0] | First Dynamic Buffer Bit<br>0: No group of message buffers exclusively for the static segment configured<br>1 to 127: Message buffers 0 to FLXAnFRMRC.FDB – 1 reserved for static segment<br>128: No dynamic message buffers configured |

**(1) FLXAnFRMRC.SPLM**

Sync Frame Payload Multiplex Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

This bit is only evaluated if the node is configured as sync node (FLXAnFRSUCC1.TXSY = 1) or for single slot mode operation (FLXAnFRSUCC1.TSM = 1).

When this bit is set to 1 message buffers 0 and 1 are dedicated for sync frame transmission with different payload data on channel A and B.

When this bit is set to 0, sync frames are transmitted from message buffer 0 with the same payload data on all channels configured. Note that the channel filter configuration for message buffer 0 resp. message buffer 1 has to be chosen accordingly.

**(2) FLXAnFRMRC.SEC**

Secure Buffers Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Not evaluated when the CC is in DEFAULT_CONFIG or CONFIG state.

For temporary unlocking, see **Section 20.3.13.4, Host Handling of Access Errors**.

$00_B$ = all buffers unlocked

Reconfiguration of message buffers enabled with numbers < FLXAnFRMRC.FFB enabled

Exception: In nodes configured for sync frame transmission or for single slot mode operation message buffer 0 (and if FLXAnFRMRC.SPLM = 1, also message buffer 1) is always locked

$01_B$ = static buffers locked, FIFO locked, limited transmission

Reconfiguration of message buffers with numbers < FLXAnFRMRC.FDB and with numbers ≥ FLXAnFRMRC.FFB locked and transmission of message buffers for static segment with numbers ≥ FLXAnFRMRC.FDB disabled

$10_B$ = all buffers locked

Reconfiguration of all message buffers locked

$11_B$ = all buffers locked, limited transmission

Reconfiguration of all message buffers locked and transmission of message buffers for static segment with numbers ≥ FLXAnFRMRC.FDB disabled

**(3) FLXAnFRMRC.LCB**

Last Configured Buffer Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

When a Static and Dynamic Buffer area is configured (FLXAnFRMRC.FDB < 128), the user should configure FLXAnFRMRC.LCB ≥ FLXAnFRMRC.FDB.

When a FIFO area is configured (FLXAnFRMRC.FFB < 128), the user should configure FLXAnFRMRC.LCB ≥ FLXAnFRMRC.FFB.

**(4) FLXAnFRMRC.FFB**

First Buffer of FIFO Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

When a Static and Dynamic Buffer area is configured (FLXAnFRMRC.FDB < 128), the user should configure FLXAnFRMRC.FFB > FLXAnFRMRC.FDB.

**(5) FLXAnFRMRC.FDB**

First Dynamic Buffer Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

### 20.2.8.2 FLXAnFRFRF — FlexRay FIFO Rejection Filter Register

The FIFO Rejection Filter defines a user specified sequence of bits to which channel, frame ID, and cycle count of the incoming frames are compared. Together with the FIFO Rejection Filter Mask this register determines whether a message is rejected by the FIFO.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0304_H$

**Value after reset:** $0180\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | RNF | RSS | CYF[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | FID[10:0] | | | | | | | | | | | CH[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.57 FLXAnFRFRF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 25 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 24 | RNF | Reject Null Frames Bit<br>0: Null frames are stored in the FIFO<br>1: Reject all null frames |
| 23 | RSS | Reject in Static Segment Bit<br>0: FIFO also used for static segment<br>1: Reject messages in static segment |
| 22 to 16 | CYF[6:0] | Cycle Counter Filter Bit<br>Cycle Counter Filter |
| 15 to 13 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 12 to 2 | FID[10:0] | Frame ID Filter Bit<br>0 to 2047: Frame ID filter values |
| 1, 0 | CH[1:0] | Channel Filter Bit<br>00: receive on both channels<br>01: receive only on channel B<br>10: receive only on channel A<br>11: no reception |

#### (1) FLXAnFRFRF.RNF

Reject Null Frames Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

If this bit is set to 1, received null frames are not stored in the FIFO.

**(2) FLXAnFRFRF.RSS**

Reject in Static Segment Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

If this bit is set to 1, the FIFO is used only for the dynamic segment.

**(3) FLXAnFRFRF.CYF**

Cycle Counter Filter Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

The 7-bit cycle counter filter determines the cycle set to which frame ID and channel rejection filter are applied. In cycles not belonging to the cycle set specified by FLXAnFRFRF.CYF, all frames are rejected. For details about the configuration of the cycle counter filter see **Section 20.3.8.2, Cycle Counter Filtering**.

**(4) FLXAnFRFRF.FID**

Frame ID Filter Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

Determines the frame ID to be rejected by the FIFO. With the additional configuration of register FLXAnFRFRFM, the corresponding frame ID filter bits are ignored, which results in further rejected frame IDs. When FLXAnFRFRFM.MFID is zero, a frame ID filter value of zero means that no frame ID is rejected.

**(5) FLXAnFRFRF.CH**

Channel Filter Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

If reception on both channels is configured, also in static segment always both frames (from channel A and B) are stored in the FIFO, even if they are identical.

### 20.2.8.3 FLXAnFRFRFM — FlexRay FIFO Rejection Filter Mask Register

The FlexRay FIFO Rejection Filter Mask specifies which of the corresponding frame ID filter bits are relevant for rejection filtering. If a bit is set to 1, it indicates that the corresponding bit in the FLXAnFRFRF register will not be considered for rejection filtering.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0308_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | MFID[10:0] | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

**Table 20.58 FLXAnFRFRFM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 12 to 2 | MFID[10:0] | Mask Frame ID Filter Bit<br>0: Corresponding frame ID filter bit is used for rejection filtering<br>1: Ignore corresponding frame ID filter bit. |
| 1, 0 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |

#### (1) FLXAnFRFRFM.MFID

Mask Frame ID Filter Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

### 20.2.8.4 FLXAnFRFCL — FlexRay FIFO Critical Level Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $030C_H$

**Value after reset:** $0000\ 0080_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CL[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.59 FLXAnFRFCL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 7 to 0 | CL[7:0] | Critical Level Bit<br>Critical Level |

#### (1) FLXAnFRFCL.CL

Critical Level Bit

The user can only write to these bits when FLXAnFRCCSV.POCS is DEFAULT_CONFIG or CONFIG.

When the receive FIFO fill level FLXAnFRFSR.RFFL is equal or greater than the critical level configured by FLXAnFRFCL.CL, the receive FIFO critical level flag FLXAnFRFSR.RFCL is set to 1.

If FLXAnFRFCL.CL is programmed to values > 128, bit FLXAnFRFSR.RFCL is never set to 1.

## 20.2.9 Message Buffer Status Registers

### 20.2.9.1 FLXAnFRMHDS — FlexRay Message Handler Status Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0310_H$

**Value after reset:** $0000\ 0080_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | MBU[6:0] | | | | | | | — | MBT[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | FMB[6:0] | | | | | | | CRAM | MFMB | FMBD | ATBF2 | ATBF1 | AMR | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R |

**Table 20.60 FLXAnFRMHDS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 30 to 24 | MBU[6:0] | Message Buffer Updated Flag<br>Message Buffer Updated |
| 23 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 22 to 16 | MBT[6:0] | Message Buffer Transmitted Flag<br>Message Buffer Transmitted |
| 15 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 14 to 8 | FMB[6:0] | Faulty Message Buffer Number Flag |
| 7 | CRAM | Clear all internal RAM's Flag<br>0: No execution of the CHI command CLEAR_RAMS<br>1: Execution of the CHI command CLEAR_RAMS ongoing |
| 6 | MFMB | Multiple Faulty Message Buffer Detection Flag<br>0 = No additional faulty message buffer.<br>1 = Additional faulty message buffer was detected while the FMBD flag is set to 1. |
| 5 | FMBD | Faulty Message Buffer Detection Flag<br>0 = No faulty message buffer.<br>1 = Message buffer referenced by FLXAnFRMHDS.FMB holds faulty data with a parity error. |
| 4 | ATBF2 | Temporary Buffer RAM B Access Error Flag<br>0 = No access error<br>1 = Access error occurred when reading the RAM B. |
| 3 | ATBF1 | Temporary Buffer RAM A Access Error Flag<br>0 = No access error.<br>1 = Access error occurred when reading the RAM A. |
| 2 | AMR | Message RAM Access Error Flag<br>0 = No access error<br>1 = Access error occurred when reading the Message RAM. |
| 1, 0 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |

**(1) FLXAnFRMHDS.MBU**

Message Buffer Updated Flag

Number of message buffer that was updated last by the CC. For this message buffer the respective ND and / or MBC flag in the FLXAnFRNDAT1/2/3/4 registers and the FLXAnFRMBSC1/2/3/4 registers are also set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(2) FLXAnFRMHDS.MBT**

Message Buffer Transmitted Flag

Number of last successfully transmitted message buffer.

If the message buffer is configured for single-shot mode, the respective TXR flag in the FLXAnFRTXRQ1/2/3/4 registers was reset to 0.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(3) FLXAnFRMHDS.FMB**

Faulty Message Buffer Number Flag

This flag indicates that an access error occurred when reading from the message buffer referenced by FLXAnFRMHDS.FMB.

The value of this flag is only valid when FLXAnFRMHDS.AMR and flag FLXAnFRMHDS.FMBD are set to 1.

This flag is not updated while the FLXAnFRMHDS.FMBD flag is 1.

This flag is cleared by the CHI command CLEAR_RAMS.

**(4) FLXAnFRMHDS.CRAM**

Internal RAM Clear Flag

This flag indicates that the CHI command CLEAR_RAMS is ongoing (all bits of the message RAM, input buffer, output buffer and nonresident buffer are written to 0).

This flag is set by the CHI command CLEAR_RAMS.

**(5) FLXAnFRMHDS.MFMB**

Multiple Faulty Message Buffer Detection Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing 1 to it.

This bit indicates that an additional faulty message buffer was detected while the FMBD flag is set.

This bit is cleared by the CHI command CLEAR_RAMS.

**(6) FLXAnFRMHDS.FMBD**

Faulty Message Buffer Detection Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing 1 to it.

This bit indicates that the message buffer referenced by FLXAnFRMHDS.FMB holds faulty data due to an access error.

This bit is cleared by the CHI command CLEAR_RAMS.

### (7) FLXAnFRMHD.ATBF2

Temporary Buffer RAM B Access Error Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing 1 to it.

This flag indicates that an access error occurred when reading the RAM B.

**CAUTION**

When this flag changes from 0 to 1, the AERR bit in the FLXAnFREIR register is set to 1. This flag can be reset by the CHI command CLEAR_RAMS.

### (8) FLXAnFRMHD.ATBF1

Temporary Buffer RAM A Access Error Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing 1 to it.

This flag indicates that an access error occurred when reading the RAM A.

**CAUTION**

When this flag changes from 0 to 1, the AERR bit in the FLXAnFREIR register is set to 1. This flag can be reset by the CHI command CLEAR_RAMS.

### (9) FLXAnFRMHDS.AMR

Message RAM Access Error Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing 1 to it.

This flag indicates that an access error occurred when reading the Message RAM.

**CAUTION**

When this flag changes from 0 to 1, the AERR bit in the FLXAnFREIR register is set to 1. This flag can be reset by the CHI command CLEAR_RAMS.

### 20.2.9.2 FLXAnFRLDTS — FlexRay Last Dynamic Transmit Slot Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0314_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | LDTB[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | LDTA[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.61 FLXAnFRLDTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 26 to 16 | LDTB[10:0] | Last Dynamic Transmission Channel B Flag |
| 15 to 11 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 10 to 0 | LDTA[10:0] | Last Dynamic Transmission Channel A Flag |

#### (1) FLXAnFRLDTS.LDTB

Last Dynamic Transmission Channel B Flag

Stores the value of vSlotCounter[B] at the time of the last frame transmission on channel B in the dynamic segment of this node.

It is updated at the end of the dynamic segment and is reset to zero if no frame was transmitted during the dynamic segment.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS of channel B.

#### (2) FLXAnFRLDTS.LDTA

Last Dynamic Transmission Channel A Flag

Stores the value of vSlotCounter[A] at the time of the last frame transmission on channel A in the dynamic segment of this node.

It is updated at the end of the dynamic segment and is reset to zero if no frame was transmitted during the dynamic segment.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS of channel A.

### 20.2.9.3 FLXAnFRFSR — FlexRay FIFO Status Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0318_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFFL[7:0] | | | | | | | | — | — | — | — | — | RFO | RFCL | RFNE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.62 FLXAnFRFSR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 15 to 8 | RFFL[7:0] | Receive FIFO Fill Level Flag |
| 7 to 3 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 2 | RFO | Receive FIFO Overrun Flag<br>0: No receive FIFO overrun detected<br>1: A receive FIFO overrun has been detected |
| 1 | RFCL | Receive FIFO Critical Level Flag<br>0: Receive FIFO below critical level<br>1: Receive FIFO critical level reached |
| 0 | RFNE | Receive FIFO Not Empty Flag<br>0: Receive FIFO is empty<br>1: Receive FIFO is not empty |

**(1) FLXAnFRFSR.RFFL**

Receive FIFO Fill Level Flag

Indicates the number of FIFO buffers filled with new data not yet read by the Host. Maximum value is 128.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(2) FLXAnFRFSR.RFO**

Receive FIFO Overrun Flag

The flag is set to 1 by the CC when a receive FIFO overrun is detected.

When a receive FIFO overrun occurs, the oldest message is overwritten with the actual received message. In addition, interrupt flag FLXAnFREIR.RFO is set to 1.

The flag is cleared by the next FIFO read access issued by the Host.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(3) FLXAnFRFSR.RFCL**

Receive FIFO Critical Level Flag

This flag is set to 1 when the receive FIFO fill level FLXAnFRFSR.RFFL is equal or greater than the critical level as configured by FLXAnFRFCL.CL.

When FLXAnFRFSR.RFCL changes from 0 to 1 bit FLXAnFRSIR.RFCL is set to 1, and if enabled, an interrupt is generated.

The flag is cleared by the CC as soon as FLXAnFRFSR.RFFL drops below FLXAnFRFCL.CL.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(4) FLXAnFRFSR.RFNE**

Receive FIFO Not Empty Flag

This flag is set to 1 by the CC when a received valid frame (data or null frame depending on rejection mask) was stored in the FIFO. In addition, interrupt flag FLXAnFRSIR.RFNE is set to 1.

The bit is reset to 0 after the Host has read all messages from the FIFO.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

### 20.2.9.4 FLXAnFRMHDF — FlexRay Message Handler Constraints Flags Register

Some constraints exist for the Message Handler regarding bus clock frequency, Message RAM configuration, and FlexRay bus traffic. To simplify software development, constraints violations are reported by setting flags in the FLXAnFRMHDF.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $031C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | WAHP | TNSB | TNSA | TBFB | TBFA | FNFB | FNFA | SNUB | SNUA |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.63 FLXAnFRMHDF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 8 | WAHP | Write Attempt to Header Partition Flag<br>0: No write attempt to header partition<br>1: Write attempt to header partition |
| 7 | TNSB | Transmission Not Started Channel B Flag<br>0: No transmission not started on channel B<br>1: Transmission not started on channel B |
| 6 | TNSA | Transmission Not Started Channel A Flag<br>0: No transmission not started on channel A<br>1: Transmission not started on channel A |
| 5 | TBFB | Temporary buffer Access Failure B Flag<br>0: No TBF B access failure<br>1: TBF B access failure |
| 4 | TBFA | Temporary buffer Access Failure A Flag<br>0: No TBF A access failure<br>1: TBF A access failure |
| 3 | FNFB | Find Sequence Not Finished Channel B Flag<br>0: No find sequence not finished for channel B<br>1: Find sequence not finished for channel B |
| 2 | FNFA | Find Sequence Not Finished Channel A Flag<br>0: No find sequence not finished for channel A<br>1: Find sequence not finished for channel A |
| 1 | SNUB | Status Not Updated Channel B Flag<br>0: No overload condition occurred when updating MBS for channel B<br>1: MBS for channel B not updated |
| 0 | SNUA | Status Not Updated Channel A Flag<br>0: No overload condition occurred when updating MBS for channel A<br>1: MBS for channel A not updated |

**(1) FLXAnFRMHDF.WAHP**

Write Attempt to Header Partition Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

Outside DEFAULT_CONFIG and CONFIG state this flag is set to 1 by the CC when the message handler tries to write message data into the header partition of the Message RAM due to faulty configuration of a message buffer. The write attempt is not executed, to protect the header partition from unintended write accesses.

When this flag changes from 0 to 1 in addition interrupt flag FLXAnFREIR.MHF is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(2) FLXAnFRMHDF.TNSB**

Transmission Not Started Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set to 1 by the CC when the Message Handler was not ready to start a scheduled transmission on channel B at the action point of the configured slot.

When this flag changes from 0 to 1 in addition interrupt flag FLXAnFREIR.MHF is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(3) FLXAnFRMHDF.TNSA**

Transmission Not Started Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set by the CC when the Message Handler was not ready to start a scheduled transmission on channel A at the action point of the configured slot.

When this flag changes from 0 to 1 in addition interrupt flag FLXAnFREIR.MHF is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(4) FLXAnFRMHDF.TBFB**

Temporary buffer Access Failure B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set to 1 by the CC when a read or write access to TBF B requested by PRT (Protocol controller) B could not complete within the available time.

When this flag changes from 0 to 1 in addition interrupt flag FLXAnFREIR.MHF is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(5) FLXAnFRMHDF.TBFA**

Temporary buffer Access Failure A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set to 1 by the CC when a read or write access to TBF A requested by PRT A could not complete within the available time.

When this flag changes from 0 to 1 in addition interrupt flag FLXAnFREIR.MHF is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(6) FLXAnFRMHDF.FNFB**

Find Sequence Not Finished Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set to 1 by the CC when the Message Handler, due to overload condition, was not able to finish a find sequence (scan of Message RAM for matching message buffer).

When this flag changes from 0 to 1 in addition interrupt flag FLXAnFREIR.MHF is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(7) FLXAnFRMHDF.FNFA**

Find Sequence Not Finished Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set to 1 by the CC when the Message Handler, due to overload condition, was not able to finish a find sequence (scan of Message RAM for matching message buffer).

When this flag changes from 0 to 1 in addition interrupt flag FLXAnFREIR.MHF is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(8) FLXAnFRMHDF.SNUB**

Status Not Updated Channel B Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set to 1 by the CC when the Message Handler, due to overload condition, was not able to update a message buffer's status (FLXAnFRMBS).

When this flag changes from 0 to 1 in addition interrupt flag FLXAnFREIR.MHF is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

**(9) FLXAnFRMHDF.SNUA**

Status Not Updated Channel A Flag

Writing 0 has no effect on the bit value.

This bit is cleared when writing a 1 to it.

This flag is set to 1 by the CC when the Message Handler, due to overload condition, was not able to update a message buffer's status (FLXAnFRMBS).

When this flag changes from 0 to 1 in addition interrupt flag FLXAnFREIR.MHF is set to 1.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

### 20.2.9.5 FLXAnFRTXRQi — FlexRay Transmission Request i (i = 1 to 4)

The four registers reflect the state of the TXR flags of all configured message buffers. The flags are evaluated for transmit buffers only. If the number of configured message buffers is less than 128, the remaining TXR flags have no meaning.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0320_H$ to <FLXn_base> + $032C_H$ (<FLXn_base> + $0320_H$ + (i – 1)*4)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TXRo[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TXRo[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.64 FLXAnFRTXRQi Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31:0 | TXRo[31:0] | Transmission Request Flag o |

#### (1) FLXAnFRTXRQi.TXRo (o = (i – 1)*32 to (i*32 – 1))

Transmission Request Flag o

If the flag is set to 1, the respective message buffer is ready for transmission respectively transmission of this message buffer is in progress.

In single-shot mode the flags are reset to 0 after transmission has completed.

This bit is cleared by the CHI command CLEAR_RAMS.

### 20.2.9.6 FLXAnFRNDATi — FlexRay New Data Register i (i = 1 to 4)

The four registers reflect the state of the ND flags of all configured message buffers. ND flags belonging to transmit buffers have no meaning. If the number of configured message buffers is less than 128, the remaining ND flags have no meaning.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0330_H$ to <FLXn_base> + $033C_H$ (<FLXn_base> + $0330_H$ + (i – 1)*4)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NDn[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NDn[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.65 FLXAnFRNDATi Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | NDn[31:0] | New Data Flag n |

#### (1) FLXAnFRNDATi.NDn (n = (i – 1)*32 to (i*32 – 1))

New Data Flag n

The flags are set to 1 when a valid received data frame matches the message buffer's filter configuration, independent of the payload length received or the payload length configured for that message buffer.

The flags are not set to 1 after reception of null frames except for message buffers belonging to the receive FIFO.

An ND flag is reset to 0 when the header section of the corresponding message buffer is reconfigured or when the data section has been transferred to the Output Buffer.

Reset when leaving CONFIG state or when entering STARTUP state.

This bit is cleared by the CHI command CLEAR_RAMS.

### 20.2.9.7 FLXAnFRMBSCi — FlexRay Message Buffer Status Changed Register i (i = 1 to 4)

The four registers reflect the state of the MBC flags of all configured message buffers. If the number of configured message buffers is less than 128, the remaining MBC flags have no meaning.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0340_H$ to <FLXn_base> + $034C_H$ (<FLXn_base> + $0340_H$ + (i – 1)*4)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBCm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.66 FLXAnFRMBSCi (i = 1 to 4) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | MBCm[31:0] | Message Buffer Status Changed Flag m |

#### (1) FLXAnFRMBSCi.MBCm (m = (i – 1)*32 to (i*32 – 1))

Message Buffer Status Changed Flag m

Indicates whether the Message Handler has changed one of the status flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB in the header section (see **Section 20.2.12.5, FLXAnFRMBS — FlexRay Message Buffer Status Register** and **Section 20.3.13.1, Header Partition** of the respective message buffer.

An MBC flag is reset to 0 when the header section of the corresponding message buffer is reconfigured or when it has been transferred to the Output Buffer.

Reset when leaving CONFIG state or when entering STARTUP state.

This bit is cleared by the CHI command CLEAR_RAMS.

## 20.2.10 Identification Registers

### 20.2.10.1 FLXAnFRPV — FlexRay Product Version Register

**Access:** This register can be read/ in 32-bit units.

**Address:** <FLXn_base> + $0000_H$

**Value after reset:** 2A10 $2300_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AID[7:0] | | | | | | | | INV[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | JID[7:0] | | | | | | | | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.67 FLXAnFRPV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | AID[7:0] | AUTOSAR ID<br>Fixed to $2A_H$ |
| 23 to 16 | INV[7:0] | Internal version<br>Release version of the FlexRay IP |
| 15 to 8 | JID[7:0] | JEDEC ID<br>Fixed to $23_H$ |
| 7 to 0 | Reserved | When read, the value after reset always is read. When writing |

#### (1) FLXAnFRPV.AID

AUTOSAR ID

#### (2) FLXAnFRPV.INV

Internal version

These bits show the release version of the FlexRay IP

The bits FLXAnFRPV.INV[7:4] show the main release number and the bits FLXAnFRPV.INV[3:0] show the sub release number.

Examples:

10h = IP release 1.0

11h = IP release 1.1

20h = IP release 2.0

#### (3) FLXAnFRPV.JID

JEDEC ID

### 20.2.11 Input Buffer

Double buffer structure consisting of Input Buffer Host and Input Buffer Shadow. While the Host can write to Input Buffer Host, the transfer to the Message RAM is done from Input Buffer Shadow. The Input Buffer holds the header and data sections to be transferred to the selected message buffer in the Message RAM. It is used to configure the message buffers in the Message RAM and to update the data sections of transmit buffers.

When updating the header section of a message buffer in the Message RAM from the Input Buffer, the Message Buffer Status as described in **Section 20.2.12.5, FLXAnFRMBS — FlexRay Message Buffer Status Register** is automatically reset to zero.

The header sections of message buffers belonging to the receive FIFO can only be (re)configured when the CC is in DEFAULT_CONFIG or CONFIG state. For those message buffers only the payload length configured and the data pointer need to be configured via FLXAnFRWRHS2.PLC and FLXAnFRWRHS3.DP. All information required for acceptance filtering is taken from the FIFO rejection filter and the FIFO rejection filter mask.

The data transfer between Input Buffer (IBF) and Message RAM is described in detail in **Section 20.3.12.2, Host access to Message RAM**.

These registers cannot be written when the input data transfer function shown in **Section 20.3.16.1, Input Data Transfer** is used and the FLXAnFRITS.ITS bit is 1.

### 20.2.11.1 FLXAnFRWRDSx — FlexRay Write Data Section Register x (x = 1 to 64)

This register holds the data words to be transferred to the data section of the specified message buffer. The number of data words written to the Message RAM is defined by the payload length configured in FLXAnFRWRHS2.PLC bit.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0400_H$ to <FLXn_base> + $04FC_H$ (<FLXn_base> + $0400_H$ + (x – 1)*4)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MD[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MD[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.68 FLXAnFRWRDSx Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | MD[31:0] | Message Data Bit |

#### (1) FLXAnFRWRDSx.MD

Message Data Bit

For information about the byte alignment of the message data in this register see **Section 20.3.17, Byte Alignment**.

**CAUTIONS**

1. In case FLXAnFRWRHS2.PLC specifies an odd payload length, the remaining message data bytes are unused.
2. When writing to FLXAnFRWRDSn, each 32-bit word has to be filled up by one 32-bit access OR two consecutive 16-bit accesses OR four consecutive 8- bit accesses before the transfer from the Input Buffer to the Message RAM is started. If not all bytes of a 32- bit word have been written by the Host (8/16-bit access only), FLXAnFRWRDSn holds partly undefined data.
   Reset by the CHI command CLEAR_RAMS.

### 20.2.11.2 FLXAnFRWRHS1 — FlexRay Write Header Section Register 1

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0500_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | MBI | TXM | PPIT | CFG | CH[1:0] | | — | CYC[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | FID[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.69 FLXAnFRWRHS1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 | MBI | Message Buffer Interrupt Bit<br>0: The corresponding message buffer interrupt is disabled<br>1: The corresponding message buffer interrupt is enabled |
| 28 | TXM | Transmission Mode Setting Bit<br>0: Continuous mode<br>1: Single-shot mode |
| 27 | PPIT | Payload Preamble Indicator Transmit Bit<br>0: Payload Preamble Indicator is set to 0<br>1: Payload Preamble Indicator is set to 1 |
| 26 | CFG | Message Buffer Direction Configuration Bit<br>0: The corresponding buffer is configured as Receive Buffer<br>1: The corresponding buffer is configured as Transmit Buffer |
| 25, 24 | CH[1:0] | Channel Filter Control Bit |
| 23 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 22 to 16 | CYC[6:0] | Cycle Code Bit |
| 15 to 11 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 10 to 0 | FID[10:0] | Frame ID Bit |

#### (1) FLXAnFRWRHS1.MBI

Message Buffer Interrupt Enable Bit

This bit enables the message buffer interrupt.

After a dedicated receive buffer has been updated by the Message Handler, flag FLXAnFRSIR.RXI and /or FLXAnFRSIR.MBSI are set to 1. After a transmission has completed flag FLXAnFRSIR.TXI is set to 1.

#### (2) FLXAnFRWRHS1.TXM

Transmission Mode Setting Bit

This bit selects transmit mode of the corresponding message buffer. For transmit mode, see **Section 20.3.9.3, Transmit Buffers**.

#### (3) FLXAnFRWRHS1.PPIT

Payload Preamble Indicator Transmit Bit

This bit is used to control the state of the Payload Preamble Indicator in transmit frames in transmit frames of the corresponding message buffer.

If the bit is set to 1 in a static message buffer, the respective message buffer holds network management information.

If the bit is set to 1 in a dynamic message buffer the first two bytes of the payload segment may be used for message ID filtering by the receiver. Message ID filtering of received FlexRay frames is not supported by the FlexRay module, but can be done by the Host.

#### (4) FLXAnFRWRHS1.CFG

Message Buffer Direction Configuration Bit

This bit is used to configure the corresponding buffer as transmit buffer or as receive buffer. For message buffers belonging to the receive FIFO the bit is not evaluated.

If the user doesn't leave at least 32bit unused area at the head of Data Partition, the user should configure the Message Buffer which Data Section is located directly following the Header partition (the last configured buffer) as transmit buffer by setting this bit to “1”.

#### (5) FLXAnFRWRHS1.CH

Channel Filter Control Bit

The 2-bit channel filtering field associated with each buffer serves as a filter for receive buffers, and as a control field for transmit buffers.

| CH[1:0] | Transmit Buffer transmit frame on | Receive Buffer store frame received from |
|---|---|---|
| 00 | No transmission | Ignore frame |
| 01 | Channel A | Channel A |
| 10 | Channel B | Channel B |
| 11 | Both channels (static segment only) | Channel A or B (store first semantically valid frame; static segment only) |

**CAUTION**

If a message buffer is configured for the dynamic segment and both bits of the channel filtering field are set to 1, no frames are transmitted resp. received frames are ignored (same function as CH = “$00_B$”)

**(6) FLXAnFRWRHS1.CYC**

Cycle Code Bit

The 7-bit cycle code determines the cycle set used for cycle counter filtering.

For details about the configuration of the cycle code **Section 20.3.8.2, Cycle Counter Filtering**.

**(7) FLXAnFRWRHS1.FID**

Frame ID Bit

Frame ID of the selected message buffer. The frame ID defines the slot number for transmission / reception of the respective message.

Message buffers with frame ID = 0 are considered as not valid.

### 20.2.11.3 FLXAnFRWRHS2 — FlexRay Write Header Section Register 2

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0504_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | PLC[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CRC[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.70 FLXAnFRWRHS2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 22 to 16 | PLC[6:0] | Payload Length Configured Bit |
| 15 to 11 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 10 to 0 | CRC[10:0] | Header CRC Bit (vRF!Header!HeaderCRC)<br>Receive Buffer: Configuration not required<br>Transmit Buffer: Header CRC is configured |

#### (1) FLXAnFRWRHS2.PLC

Payload Length Configured Bit

Length of data section (number of 2-byte words) as configured by the Host.

During static segment the static frame payload length as configured by FLXAnFRMHDC.SFDL defines the payload length for all static frames. If the payload length configured by FLXAnFRWRHS2.PLC is shorter than this value padding bytes are inserted to ensure that frames have proper physical length. The padding pattern is "$0000_H$" (see **Section 20.3.9.3, Transmit Buffers**).

#### (2) FLXAnFRWRHS2.CRC

Header CRC Bit (vRF!Header!HeaderCRC)

Setting of the receive buffer is not required.

Transmitting of the message buffer needs the header CRC calculation and setting.

For calculation of the header CRC the payload length of the frame send on the bus has to be considered. In static segment the payload length of all frames is configured by FLXAnFRMHDC.SFDL.

### 20.2.11.4 FLXAnFRWRHS3 — FlexRay Write Header Section Register 3

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0508_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | DP[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.71 FLXAnFRWRHS3 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 10 to 0 | DP[10:0] | Data Pointer Bit |

#### (1) FLXAnFRWRHS3.DP

Data Pointer Bit

Configures the pointer to the first 32-bit word of the data section of the addressed message buffer in the Message RAM.

### 20.2.11.5 FLXAnFRIBCM — FlexRay Input Buffer Command Mask Register

Configures how the message buffer in the Message RAM selected by register FLXAnFRIBCR is updated. When IBF Host and IBF Shadow are swapped, also mask bits FLXAnFRIBCM.LHSH, FLXAnFRIBCM.LDSH, and FLXAnFRIBCM.STXRH are swapped with bits FLXAnFRIBCM.LHSS, FLXAnFRIBCM.LDSS, and FLXAnFRIBCM.STXRS.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0510_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | STXRS | LDSS | LHSS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | STXRH | LDSH | LHSH |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 20.72 FLXAnFRIBCM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 18 | STXRS | Set Transmission Request Shadow Flag<br>0: Reset TXR flag<br>1: Set TXR flag, transmit buffer released for transmission (operation ongoing or finished) |
| 17 | LDSS | Load Data Section Shadow Flag<br>0: Data section is not updated<br>1: Data section selected for transfer from Input Buffer to the Message RAM (transfer ongoing or finished) |
| 16 | LHSS | Load Header Section Shadow Flag<br>0: Header section is not updated<br>1: Header section selected for transfer from Input Buffer to the Message RAM (transfer ongoing or finished) |
| 15 to 3 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 2 | STXRH | Set Transmission Request Host Bit<br>0: Reset TXR flag<br>1: Set TXR flag, transmit buffer released for transmission |
| 1 | LDSH | Load Data Section Host Bit<br>0: Data section is not updated<br>1: Data section selected for transfer from Input Buffer to the Message RAM |
| 0 | LHSH | Load Header Section Host Bit<br>0: Header section is not updated<br>1: Header section selected for transfer from Input Buffer to the Message RAM |

**(1) FLXAnFRIBCM.STXRS**

Set Transmission Request Shadow Flag

**(2) FLXAnFRIBCM.LDSS**

Load Data Section Shadow Flag

**(3) FLXAnFRIBCM.LHSS**

Load Header Section Shadow Flag

**(4) FLXAnFRIBCM.STXRH**

Set Transmission Request Host Bit

If this bit is set to 1, the TXR flag for the selected message buffer is set in the FLXAnFRTXRQ1/2/3/4 registers to release the message buffer for transmission. In single-shot mode the flag is cleared by the CC after transmission has completed.

TXR is evaluated for transmit buffers only.

**(5) FLXAnFRIBCM.LDSH**

Set Load Data Section Host Bit

**(6) FLXAnFRIBCM.LHSH**

Set Load Header Section Host Bit

### 20.2.11.6 FLXAnFRIBCR — FlexRay Input Buffer Command Request Register

When the Host writes the number of the target message buffer in the Message RAM to FLXAnFRIBCR.IBRH, IBF Host and IBF Shadow are swapped. In addition the message buffer numbers stored under FLXAnFRIBCR.IBRH and FLXAnFRIBCR.IBRS are also swapped (see **Section 20.3.12.2 (1), Data Transfer from Input Buffer to Message RAM**).

With this write operation the FLXAnFRIBCR.IBSYS is set to 1. The Message Handler then starts to transfer the contents of IBF Shadow to the message buffer in the Message RAM selected by FLXAnFRIBCR.IBRS.

While the Message Handler transfers the data from IBF Shadow to the target message buffer in the Message RAM, the Host may write the next message into the IBF Host. After the transfer between IBF Shadow and the Message RAM has completed, FLXAnFRIBCR.IBSYS is set back to 0 and the next transfer to the Message RAM may be started by the Host by writing the respective target message buffer number to FLXAnFRIBCR.IBRH.

If a write access to FLXAnFRIBCR.IBRH occurs while FLXAnFRIBCR.IBSYS is 1, FLXAnFRIBCR.IBSYH is set to 1. After completion of the ongoing data transfer from IBF Shadow to the Message RAM, IBF Host and IBF Shadow are swapped, FLXAnFRIBCR.IBSYH is reset to 0. FLXAnFRIBCR.IBSYS remains set to 1, and the next transfer to the Message RAM is started. In addition the message buffer numbers stored under FLXAnFRIBCR.IBRH and FLXAnFRIBCR.IBRS are also swapped.

Any write access to an Input Buffer register while both FLXAnFRIBCR.IBSYS and FLXAnFRIBCR.IBSYH are set to 1 will cause the error flag FLXAnFREIR.IIBA to be set to 1.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0514_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IBSYS | — | — | — | — | — | — | — | — | IBRS[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IBSYH | — | — | — | — | — | — | — | — | IBRH[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.73 FLXAnFRIBCR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | IBSYS | Input Buffer Busy Shadow Flag<br>0: Transfer between IBF Shadow and Message RAM completed<br>1: Transfer between IBF Shadow and Message RAM in progress |
| 30 to 23 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 22 to 16 | IBRS[6:0] | Input Buffer Request Shadow Flag |

Table 20.73 FLXAnFRIBCR Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | IBSYH | Input Buffer Busy Host Flag<br>0: No request pending<br>1: Request while transfer between IBF Shadow and Message RAM in progress |
| 14 to 7 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 6 to 0 | IBRH[6:0] | Input Buffer Request Host Bit |

#### (1) FLXAnFRIBCR.IBSYS

Input Buffer Busy Shadow Flag

Set to 1 after writing FLXAnFRIBCR.IBRH.

This bit indicates transmitting between the IBF Shadow and the Message RAM is ongoing.

When the transfer between IBF Shadow and the Message RAM has completed, FLXAnFRIBCR.IBSYS is set back to 0.

#### (2) FLXAnFRIBCR.IBRS

Input Buffer Request Shadow Flag

Number of the target message buffer actually updated / lately updated.

#### (3) FLXAnFRIBCR.IBSYH

Input Buffer Busy Host Flag

Set to 1 by writing FLXAnFRIBCR.IBRH while FLXAnFRIBCR.IBSYS is still 1.

This bit indicates transmitting between the IBF Shadow and the Message RAM is ongoing.

After the ongoing transfer between IBF Shadow and the Message RAM has completed, the FLXAnFRIBCR.IBSYH is set back to 0.

#### (4) FLXAnFRIBCR.IBRH

Input Buffer Request Host Bit

Selects the target message buffer in the Message RAM for data transfer from Input Buffer.

## 20.2.12 Output Buffer

Double buffer structure consisting of Output Buffer Host and Output Buffer Shadow. Used to read out message buffers from the Message RAM. While the Host can read from Output Buffer Host, the Message Handler transfers the selected message buffer from Message RAM to Output Buffer Shadow. The data transfer between Message RAM and Output Buffer (OBF) is described in **Section 20.3.12.2 (2), Data Transfer from Message RAM to Output Buffer**.

These registers cannot be written when the output data transfer function shown in **Section 20.3.16.2, Output Data Transfer**, in Output Data Transfer is used and the FLXAnFROTS.OTS bit is 1.

### 20.2.12.1 FLXAnFRRDDSx — FlexRay Read Data Section Register x (x = 1 to 64)

Holds the data words read from the data section of the addressed message buffer. This register holds the data words to be transferred to the data section of the specified message buffer. The number of data words (DWn) read from the Message RAM is defined by the payload length configured in FLXAnFRRDHS2.PLC bit.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0600_H$ to <FLXn_base> + $06FC_H$ (<FLXn_base> + $0600_H$ + (x – 1)*4)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MD[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MD[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.74 FLXAnFRRDDSn Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | MD[31:0] | Message Data |

#### (1) FLXAnFRRDDSn.MD

Message Data Flag

For information about the byte alignment of the data words in this register see **Section 20.3.17, Byte Alignment**.

**CAUTION**

In case FLXAnFRWRHS2.PLC specifies an odd payload length, the remaining message data bytes are unused.

Reset by the CHI command CLEAR_RAMS.

### 20.2.12.2 FLXAnFRRDHS1 — FlexRay Read Header Section Register 1

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0700_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | MBI | TXM | PPIT | CFG | CH[1:0] | | — | CYC[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | FID[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.75 FLXAnFRRDHS1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 | MBI | Message Buffer Interrupt Flag |
| 28 | TXM | Transmission Mode Flag |
| 27 | PPIT | Payload Preamble Indicator Transmit Flag |
| 26 | CFG | Message Buffer Direction Configuration Flag |
| 25, 24 | CH[1:0] | Channel Filter Control Flag |
| 23 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 22 to 16 | CYC[6:0] | Cycle Code |
| 15 to 11 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 10 to 0 | FID[10:0] | Frame ID |

#### (1) FLXAnFRRDHS1.MBI

Message Buffer Interrupt Flag

Values as configured by the Host via FLXAnFRWRHS1.MBI.

In case that the message buffer read from the Message RAM belongs to the receive FIFO this bit is set to 0.

#### (2) FLXAnFRRDHS1.TXM

Transmission Mode Flag

Values as configured by the Host via FLXAnFRWRHS1.TXM.

In case that the message buffer read from the Message RAM belongs to the receive FIFO this bit is set to 0.

**(3) FLXAnFRRDHS1.PPIT**

Payload Preamble Indicator Transmit Flag

Values as configured by the Host via FLXAnFRWRHS1.PPIT.

In case that the message buffer read from the Message RAM belongs to the receive FIFO this bit is set to 0.

**(4) FLXAnFRRDHS1.CFG**

Message Buffer Direction Configuration Flag

Values as configured by the Host via FLXAnFRWRHS1.CFG.

In case that the message buffer read from the Message RAM belongs to the receive FIFO this bit is set to 0.

**(5) FLXAnFRRDHS1.CH**

Channel Filter Control Flag

Values as configured by the Host via FLXAnFRWRHS1.CH.

In case that the message buffer read from the Message RAM belongs to the receive FIFO these bits are set to 0.

**(6) FLXAnFRRDHS1.CYC**

Cycle Code

Values as configured by the Host via FLXAnFRWRHS1.CYC.

In case that the message buffer read from the Message RAM belongs to the receive FIFO these bits are set to 0.

**(7) FLXAnFRRDHS1.FID**

Frame ID

Values as configured by the Host via FLXAnFRWRHS1.FID

In case that the message buffer read from the Message RAM belongs to the receive FIFO these bits are holding the received frame ID.

### 20.2.12.3 FLXAnFRRDHS2 — FlexRay Read Header Section Register 2

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0704_H$

**Value after reset:** $0000\ 0000_H$

**CAUTION**

For message buffers belonging to the Static Buffer area or Static and Dynamic Buffer area FLXAnFRWRHS2 is updated from data frames only.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | PLR[6:0] | | | | | | | — | PLC[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CRC[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.76 FLXAnFRRDHS2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 30 to 24 | PLR[6:0] | Payload Length Received Flag (vRF!Header!Length) |
| 23 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 22 to 16 | PLC[6:0] | Payload Length Configured Flag |
| 15 to 11 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 10 to 0 | CRC[10:0] | Header CRC Flag (vRF!Header!HeaderCRC) |

#### (1) FLXAnFRRDHS2.PLR

Payload Length Received Flag (vRF!Header!Length)

Payload length (vRF!Header!Length) value updated from received data frames (exception: if message buffer belongs to the receive FIFO FLXAnFRRDSH2.PLR is also updated from received null frames).

#### (2) FLXAnFRRDHS2.PLC

Payload Length Configured Flag

Length of data section (number of 2-byte words) as configured by the Host.

#### (3) FLXAnFRRDHS2.CRC

Header CRC Flag (vRF!Header!HeaderCRC)

Receive Buffer: Header CRC (vRF!Header!HeaderCRC) updated from received data frames

Transmit Buffer: Header CRC configured by the Host

### (4) Data storage

When a message is stored into a message buffer the following behavior with respect to payload length received and payload length configured is implemented:

FLXAnFRRDHS2.PLR > FLXAnFRRDHS2.PLC:

The payload data stored in the message buffer is truncated to the payload length configured if FLXAnFRRDHS2.PLC even or else truncated to FLXAnFRRDHS2.PLC + 1.

FLXAnFRRDHS2.PLR <= FLXAnFRRDHS2.PLC:

The received payload data is stored into the message buffers data section. The remaining data bytes of the data section as configured by FLXAnFRRDHS2.PLC are filled with undefined data.

FLXAnFRRDHS2.PLR = zero:

The message buffer’s data section is filled with undefined data

FLXAnFRRDHS2.PLC = zero:

Message buffer has no data section configured. No data is stored into the message buffer's data section.

**CAUTIONS**

1. The Message RAM is organized in 4-byte words. When received data is stored into a message buffer’s data section, the number of 2-byte data words written into the message buffer is FLXAnFRRDHS2.PLC rounded to the next even value.
2. FLXAnFRRDHS2.PLC should be configured identical for all message buffers belonging to the receive FIFO. Header 2 is updated from data frames only.
   For message buffers belonging to the Static Buffer area or Static and Dynamic Buffer area FLXAnFRWRHS2 is updated from data frames only.

### 20.2.12.4 FLXAnFRRDHS3 — FlexRay Read Header Section Register 3

**CAUTION**

For message buffers belonging to the Static Buffer area or Static and Dynamic Buffer area FLXAnFRWRHS3 is updated from data frames only.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0708_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | RES | PPI | NFI | SYN | SFI | RCI | — | — | RCC[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | DP[10:0] | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.77 FLXAnFRRDHS3 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 | RES | Reserved Bit Indicator Flag (vRF!Header!Reserved) |
| 28 | PPI | Payload Preamble Indicator (vRF!Header!PPIndicator) |
| 27 | NFI | Null Frame Indicator Flag (vRF!Header!NFIndicator)<br>0: Up to now no data frame has been stored into the respective message buffer<br>1: At least one data frame has been stored into the respective message buffer |
| 26 | SYN | Sync Frame Indicator Flag (vRF!Header!SyFIndicator)<br>0: The received frame is not a sync frame<br>1: The received frame is a sync frame |
| 25 | SFI | Startup Frame Indicator Flag (vRF!Header!SuFIndicator)<br>0: The received frame is not a startup frame<br>1: The received frame is a startup frame |
| 24 | RCI | Received on Channel Indicator Flag (vSS!Channel)<br>0: Frame received on channel B<br>1: Frame received on channel A |
| 23, 22 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 21 to 16 | RCC[5:0] | Receive Cycle Counter (vRF!Header!CycleCount) |
| 15 to 11 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 10 to 0 | DP[10:0] | Data Pointer Flag |

**(1) FLXAnFRRDHS3.RES**

Reserved Bit Flag (vRF!Header!Reserved)

Reflects the state of the received reserved bit. The reserved bit is transmitted as 0.

**(2) FLXAnFRRDHS3.PPI**

Payload Preamble Indicator Flag (vRF!Header!PPIndicator)

The payload preamble indicator defines whether a network management vector or message ID is contained within the payload segment of the received frame.

0 = The payload segment of the received frame does not contain a network management vector nor a message ID

1 = Static segment: Network management vector in the first part of the payload
Dynamic segment: Message ID in the first part of the payload

**(3) FLXAnFRRDHS3.NFI**

Null Frame Indicator Flag (vRF!Header!NFIndicator)

Is set to 1 after storage of the first received data frame.

**(4) FLXAnFRRDHS3.SYN**

Sync Frame Indicator Flag (vRF!Header!SyFIndicator)

A sync frame is marked by the sync frame indicator.

**(5) FLXAnFRRDHS3.SFI**

Startup Frame Indicator Flag (vRF!Header!SuFIndicator)

A startup frame is marked by the startup frame indicator.

**(6) FLXAnFRRDHS3.RCI**

Received on Channel Indicator Flag (vSS!Channel)

Indicates the channel from which the received data frame was taken to update the respective receive buffer.

**(7) FLXAnFRRDHS3.RCC**

Receive Cycle Counter (vRF!Header!CycleCount)

Cycle counter value updated from received data frame.

**(8) FLXAnFRRDHS3.DP**

Data Pointer Flag

Pointer to the first 32-bit word of the data section of the addressed message buffer in the Message RAM.

The bit value is the same as that set in the FLXAnFRWRHS3.DP bit.

### 20.2.12.5 FLXAnFRMBS — FlexRay Message Buffer Status Register

The message buffer status is updated by the CC with respect to the assigned channel(s) latest at the end of the slot following the slot assigned to the message buffer.

The flags are updated only when the CC is in NORMAL_ACTIVE or NORMAL_PASSIVE state.

If only one channel (A or B) is assigned to a message buffer, the channel-specific status flags of the other channel are written to zero. If both channels are assigned to a message buffer, the channel-specific status flags of both channels are updated.

The message buffer status is updated only when the slot counter reached the configured frame ID and when the cycle counter filter matched. When the Host updates a message buffer via Input Buffer, all FLXAnFRMBS flags are reset to zero independent of which FLXAnFRIBCM bits are set or not.

For details about receive / transmit filtering see **Section 20.3.8, Filtering and Masking**, **Section 20.3.9, Transmit Process** and **Section 20.3.10, Receive Process**.

Whenever the Message Handler changes one of the flags FLXAnFRMBS.VFRA, FLXAnFRMBS.VFRB, FLXAnFRMBS.SEOA, FLXAnFRMBS.SEOB, FLXAnFRMBS.CEOA, FLXAnFRMBS.CEOB, FLXAnFRMBS.SVOA, FLXAnFRMBS.SVOB, FLXAnFRMBS.TCIA, FLXAnFRMBS.TCIB, FLXAnFRMBS.ESA, FLXAnFRMBS.ESB, FLXAnFRMBS.MLST, FLXAnFRMBS.FTA, FLXAnFRMBS.FTB the respective message buffer's MBC flag in registers FLXAnFRMBSC1/2/3/4 is set.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $070C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | RESS | PPIS | NFIS | SYNS | SFIS | RCIS | — | — | CCS[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FTB | FTA | — | MLST | ESB | ESA | TCIB | TCIA | SVOB | SVOA | CEOB | CEOA | SEOB | SEOA | VFRB | VFRA |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.78 FLXAnFRMBS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 | RESS | Reserved Bit Status Flag (vRF!Header!Reserved) |
| 28 | PPIS | Payload Preamble Indicator Status Flag (vRF!Header!PPIndicator)<br>0: PPI indicator set to 0<br>1: PPI indicator set to 1 |
| 27 | NFIS | Null Frame Indicator Status Flag (vRF!Header!NFIndicator)<br>0: Received frame is a null frame<br>1: Received frame is not a null frame |
| 26 | SYNS | Sync Frame Indicator Status Flag (vRF!Header!SyFIndicator)<br>0: No sync frame received<br>1: The received frame is a sync frame |

Table 20.78 FLXAnFRMBS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 25 | SFIS | Startup Frame Indicator Status Flag (vRF!Header!SuFIndicator)<br>0: No startup frame received<br>1: The received frame is a startup frame |
| 24 | RCIS | Received on Channel Indicator Status Flag (vSS!Channel)<br>0: Frame received on channel B<br>1: Frame received on channel A |
| 23, 22 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 21 to 16 | CCS[5:0] | Cycle Count Status Flag |
| 15 | FTB | Frame Transmitted on Channel B Flag<br>0: No data frame transmitted on channel B<br>1: Data frame transmitted on channel B |
| 14 | FTA | Frame Transmitted on Channel A Flag<br>0: No data frame transmitted on channel A<br>1: Data frame transmitted on channel A |
| 13 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 12 | MLST | Message Lost Flag<br>0: No message lost<br>1: Unprocessed message was overwritten |
| 11 | ESB | Empty Slot Channel B Flag<br>0: Bus activity detected in the assigned slot on channel B<br>1: No bus activity detected in the assigned slot on channel B |
| 10 | ESA | Empty Slot Channel A Flag<br>0: Bus activity detected in the assigned slot on channel A<br>1: No bus activity detected in the assigned slot on channel A |
| 9 | TCIB | Transmission Conflict Indication Channel B Flag (vSS!TxConflictB)<br>0: No transmission conflict occurred on channel B<br>1: Transmission conflict occurred on channel B |
| 8 | TCIA | Transmission Conflict Indication Channel A Flag (vSS!TxConflictA)<br>0: No transmission conflict occurred on channel A<br>1: Transmission conflict occurred on channel A |
| 7 | SVOB | Slot Boundary Violation Observed on Channel B Flag (vSS!BViolationB)<br>0: No slot boundary violation observed on channel B<br>1: Slot boundary violation observed on channel B |
| 6 | SVOA | Slot Boundary Violation Observed on Channel A Flag (vSS!BViolationA)<br>0: No slot boundary violation observed on channel A<br>1: Slot boundary violation observed on channel A |
| 5 | CEOB | Content Error Observed on Channel B Flag (vSS!ContentErrorB)<br>0: No content error observed on channel B<br>1: Content error observed on channel B |
| 4 | CEOA | Content Error Observed on Channel A Flag (vSS!ContentErrorA)<br>0: No content error observed on channel A<br>1: Content error observed on channel A |
| 3 | SEOB | Syntax Error Observed on Channel B Flag (vSS!SyntaxErrorB)<br>0: No syntax error observed on channel B<br>1: Syntax error observed on channel B |
| 2 | SEOA | Syntax Error Observed on Channel A Flag (vSS!SyntaxErrorA)<br>0: No syntax error observed on channel A<br>1: Syntax error observed on channel A |
| 1 | VFRB | Valid Frame Received on Channel B (vSS!ValidFrameB)<br>0: No valid frame received on channel B<br>1: Valid frame received on channel B |
| 0 | VFRA | Valid Frame Received on Channel A Flag (vSS!ValidFrameA)<br>0: No valid frame received on channel A<br>1: Valid frame received on channel A |

### (1) FLXAnFRMBS.RESS

Reserved Bit Status Flag (vRF!Header!Reserved)

Reflects the state of the received reserved bit. The reserved bit is transmitted as 0.

For receive buffers (FLXAnFRWRHS1.CFG = 0) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

### (2) FLXAnFRMBS.PPIS

Payload Preamble Indicator Status Flag (vRF!Header!PPIndicator)

The payload preamble indicator defines whether a network management vector or message ID is contained within the payload segment of the received frame.

For receive buffers (FLXAnFRWRHS1.CFG = 0) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

0 = PPI indicator set to 0
The payload segment of the received frame does not contain a network management vector or a message ID

1 = PPI indicator set to 1
Static segment: Network management vector at the beginning of the payload
Dynamic segment: Message ID at the beginning of the payload

### (3) FLXAnFRMBS.NFIS

Null Frame Indicator Status Flag (vRF!Header!NFIndicator)

If set to 0 the payload segment of the received frame contains no usable data.

For receive buffers (FLXAnFRWRHS1.CFG = 0) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

### (4) FLXAnFRMBS.SYNS

Sync Frame Indicator Status Flag (vRF!Header!SyFIndicator)

A sync frame is marked by the sync frame indicator.

For receive buffers (FLXAnFRWRHS1.CFG = 0) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

### (5) FLXAnFRMBS.SFIS

Startup Frame Indicator Status Flag (vRF!Header!SuFIndicator)

The startup frame indicator specifies a startup frame.

For receive buffers (FLXAnFRWRHS1.CFG = 0) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

#### (6) FLXAnFRMBS.RCIS

Received on Channel Indicator Status Flag (vSS!Channel)

Indicates the channel on which the frame was received.

For receive buffers (FLXAnFRWRHS1.CFG = 0) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

#### (7) FLXAnFRMBS.CCS

Cycle Count Status Flag

Actual cycle count when status was updated.

#### (8) FLXAnFRMBS.FTB

Frame Transmitted on Channel B Flag

Indicates that this node has transmitted a data frame in the configured slot on channel B.

**CAUTION**

The FlexRay protocol specification requires that FLXAnFRMBS.FTB can only be reset by the Host. Therefore the Cycle Count Status FLXAnFRMBS.CCS for this bit is only valid for the cycle where the bit is set to 1.

#### (9) FLXAnFRMBS.FTA

Frame Transmitted on Channel A Flag

Indicates that this node has transmitted a data frame in the configured slot on channel A.

**CAUTION**

The FlexRay protocol specification requires that FLXAnFRMBS.FTA can only be reset by the Host. Therefore the Cycle Count Status FLXAnFRMBS.CCS for this bit is only valid for the cycle where this bit is set to 1.

#### (10) FLXAnFRMBS.MLST

Message Lost Flag

The flag is set in case the Host did not read the message before the message buffer was updated from a received data frame.

Not affected by reception of null frames except for message buffers belonging to the receive FIFO. The flag is reset to 0 by a Host write to the message buffer via IBF or when a new message is stored into the message buffer after the message buffers ND flag was reset to 0 by reading out the message buffer via OBF.

#### (11) FLXAnFRMBS.ESB

Empty Slot Channel B Flag

In an empty slot, there is no activity on the bus. This means that any frame transmission is not detected. This state can be checked in static and dynamic slots.

**(12) FLXAnFRMBS.ESA**

Empty Slot Channel A Flag

In an empty slot, there is no activity on the bus. This means that any frame transmission is not detected. This state can be checked in static and dynamic slots.

**(13) FLXAnFRMBS.TCIB**

Transmission Conflict Indication Channel B Flag (vSS!TxConflictB)

A transmission conflict indication is set to 1 if a transmission conflict has occurred on channel B.

**(14) FLXAnFRMBS.TCIA**

Transmission Conflict Indication Channel A Flag (vSS!TxConflictA)

A transmission conflict indication is set if a transmission conflict has occurred on channel A.

**(15) FLXAnFRMBS.SVOB**

Slot Boundary Violation Observed on Channel B Flag (vSS!BViolationB)

A slot boundary violation (channel active at the start or at the end of the assigned slot) was observed on channel B.

**(16) FLXAnFRMBS.SVOA**

Slot Boundary Violation Observed on Channel A Flag (vSS!BViolationA)

A slot boundary violation (channel active at the start or at the end of the assigned slot) was observed on channel A.

**(17) FLXAnFRMBS.CEOB**

Content Error Observed on Channel B Flag (vSS!ContentErrorB)

A content error was observed in the assigned slot on channel B.

**(18) FLXAnFRMBS.CEOA**

Content Error Observed on Channel A Flag (vSS!ContentErrorA)

A content error was observed in the assigned slot on channel A.

**(19) FLXAnFRMBS.SEOB**

Syntax Error Observed on Channel B Flag (vSS!SyntaxErrorB)

A syntax error was observed in the assigned slot on channel B.

**(20) FLXAnFRMBS.SEOA**

Syntax Error Observed on Channel A Flag (vSS!SyntaxErrorA)

A syntax error was observed in the assigned slot on channel A.

**(21) FLXAnFRMBS.VFRB**

Valid Frame Received on Channel B Flag (vSS!ValidFrameB)

A valid frame indication is set if a valid frame was received on channel B.

**(22) FLXAnFRMBS.VFRA**

Valid Frame Received on Channel A Flag (vSS!ValidFrameA)

A valid frame indication is set if a valid frame was received on channel A.

### 20.2.12.6 FLXAnFROBCM — FlexRay Output Buffer Command Mask Register

Configures how the Output Buffer is updated from the message buffer in the Message RAM selected by FLXAnFROBCR.OBRS.

Mask bits FLXAnFROBCM.RDSS and FLXAnFROBCM.RHSS are copied to the register internal storage when a Message RAM transfer is requested by FLXAnFROBCR.REQ.

When OBF Host and OBF Shadow are swapped, mask bits FLXAnFROBCM.RDSH and FLXAnFROBCM.RHSH are swapped with the register internal storage to keep them attached to the respective Output Buffer transfer.

The data transfer between Output Buffer and Message RAM is described in detail in **Section 20.3.12.2 (2), Data Transfer from Message RAM to Output Buffer**.

**CAUTION**

After the transfer of the header section from the Message RAM to OBF Shadow has completed, the message buffer status changed flag MBC of the selected message buffer in the FLXAnFRMBSC1/2/3/4 registers is cleared. After the transfer of the data section from the Message RAM to OBF Shadow has completed, the new data flag ND of the selected message buffer in the FLXAnFRNDAT1/2/3/4 registers is cleared.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0710_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RDSH | RHSH |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RDSS | RHSS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 20.79 FLXAnFROBCM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 17 | RDSH | Read Data Section Host Flag<br>0: Data section is not read<br>1: Data section selected for transfer from Message RAM to Output Buffer |
| 16 | RHSH | Read Header Section Host Flag<br>0: Header section is not read<br>1: Header section selected for transfer from Message RAM to Output Buffer |
| 15 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | RDSS | Read Data Section Shadow Bit<br>0: Data section is not read<br>1: Data section selected for transfer from Message RAM to Output Buffer |
| 0 | RHSS | Read Header Section Shadow Bit<br>0: Header section is not read<br>1: Header section selected for transfer from Message RAM to Output Buffer |

**(1) FLXAnFROBCM.RDSH**

Read Data Section Host Flag

**(2) FLXAnFROBCM.RHSH**

Read Header Section Host Flag

**(3) FLXAnFROBCM.RDSS**

Read Data Section Shadow Bit

**(4) FLXAnFROBCM.RHSS**

Read Header Section Shadow Bit

### 20.2.12.7 FLXAnFROBCR — FlexRay Output Buffer Command Request Register

After setting bit FLXAnFROBCR.REQ to 1 while FLXAnFROBCR.OBSYS is 0, FLXAnFROBCR.OBSYS is automatically set to 1, FLXAnFROBCR.OBRS is copied to the register internal storage, mask bits FLXAnFROBCM.RDSS and FLXAnFROBCM.RHSS are copied to register FLXAnFROBCM internal storage, and the transfer of the message buffer selected by FLXAnFROBCR.OBRS from the Message RAM to OBF Shadow is started. When the transfer between the Message RAM and OBF Shadow has completed, this is signaled by setting FLXAnFROBCM.OBSYS back to 0.

By setting bit FLXAnFROBCR.VIEW to 1 while FLXAnFROBCR.OBSYS is 0, OBF Host and OBF Shadow are swapped. Additionally mask bits FLXAnFROBCM.RDSH and FLXAnFROBCM.RHSH are swapped with the register FLXAnFROBCM internal storage to keep them attached to the respective Output Buffer transfer. FLXAnFROBCR.OBRH signals the number of the message buffer currently accessible by the Host.

If bits FLXAnFROBCR.REQ and FLXAnFROBCR.VIEW are set to 1 with the same write access while FLXAnFROBCR.OBSYS is 0, FLXAnFROBCR.OBSYS is automatically set to 1 and OBF Shadow and OBF Host are swapped. Additionally mask bits FLXAnFROBCM.RDSH and FLXAnFROBCM.RHSH are swapped with the registers internal storage to keep them attached to the respective Output Buffer transfer. Afterwards FLXAnFROBCR.OBRS is copied to the register internal storage, and the transfer of the selected message buffer from the Message RAM to OBF Shadow is started. While the transfer is ongoing the Host can read the message buffer transferred by the previous transfer from OBF Host. When the current transfer between Message RAM and OBF Shadow has completed, this is signaled by setting FLXAnFROBCR.OBSYS back to 0.

Any write access to FLXAnFROBCR[15:8] while FLXAnFROBCR.OBSYS is set to 1 will cause the error flag FLXAnFREIR.IOBA to be set to 1. In this case, this write access has no effect and the Output Buffer will not be changed.

The data transfer between Output Buffer and Message RAM is described in detail in **Section 20.3.12.2 (2), Data Transfer from Message RAM to Output Buffer**.

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0714_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | OBRH[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OBSYS | — | — | — | — | — | REQ | VIEW | — | OBRS[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Table 20.80 FLXAnFROBCR Register Contents

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 22 to 16 | OBRH[6:0] | Output Buffer Request Host Flag |
| 15 | OBSYS | Output Buffer Busy Shadow Flag<br>0: No transfer in progress<br>1: Transfer between Message RAM and OBF Shadow in progress |
| 14 to 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 9 | REQ | Request Message RAM Transfer Bit<br>0: No request<br>1: Transfer to OBF Shadow requested |
| 8 | VIEW | View Shadow Buffer Bit<br>0: No action<br>1: Swap OBF Shadow and OBF Host |
| 7 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 6 to 0 | OBRS[6:0] | Output Buffer Request Shadow Bit |

### (1) FLXAnFROBCR.OBRH

Output Buffer Request Host Flag

Number of message buffer currently accessible by the Host via FLXAnFRRDHS[1…3], FLXAnFRMBS, and FLXAnFRRDDS[1…64].

By writing FLXAnFROBCR.VIEW to 1 OBF Shadow and OBF Host are swapped and the transferred message buffer is accessible by the Host.

### (2) FLXAnFROBCR.OBSYS

Output Buffer Busy Shadow Flag

Set to 1 after setting bit FLXAnFROBCR.REQ. When the transfer between the Message RAM and OBF Shadow has completed, FLXAnFROBCR.OBSYS is set back to 0.

### (3) FLXAnFROBCR.REQ

Request Message RAM Transfer Bit

Only writeable while FLXAnFROBCR.OBSYS = 0.

Requests transfer of message buffer addressed by FLXAnFROBCR.OBRS from Message RAM to OBF Shadow.

### (4) FLXAnFROBCR.VIEW

View Shadow Buffer Bit

Only writeable while FLXAnFROBCR.OBSYS = 0.

Toggles between OBF Shadow and OBF Host.

### (5) FLXAnFROBCR.OBRS

Output Buffer Request Shadow Bit

Only writeable while FLXAnFROBCR.OBSYS = 0.

Number of source message buffer to be transferred from the Message RAM to OBF Shadow.

If the number of the first message buffer of the receive FIFO is written to this register the Message Handler transfers the message buffer addressed by the GET Index (GIDX, see **Section 20.3.11, FIFO Function**) to OBF Shadow.

## 20.2.13 Data Transfer Control Register

### 20.2.13.1 FLXAnFRITC — FlexRay Input Transfer Configuration Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0800_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | ITM[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | IQEIE | IQFIE | — | — | — | — | — | — | IQHR | ITE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

**Table 20.81 FLXAnFRITC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 22 to 16 | ITM[6:0] | Input queue Table Max Bit<br>These bits configure the number of entries in the input pointer table the input buffer handler is capable to maintain in the input queue. |
| 15 to 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 9 | IQEIE | Input Queue Empty Interrupt Enable Bit<br>0: Disabled<br>1: Enabled |
| 8 | IQFIE | Input Queue Full Interrupt Enable Bit<br>0: Disabled<br>1: Enabled |
| 7 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | IQHR | Input Queue Halt Request Bit<br>0: Input queue run request<br>1: Input queue halt request |
| 0 | ITE | Input Transfer Enable Bit<br>0: Operation Disable request<br>1: Operation Enable request |

**(1) FLXAnFRITC.ITM**

Input queue Table Max Bit

The user can only write to this bit when FLXAnFRITS.ITS is 0.

These bits configure the number of entries in the input pointer table the input buffer handler is capable to maintain in the input queue.

Valid values are $00_H$ (1 queue entry) to $7F_H$ (128 queue entries).

Note that each entry requires two long words in the input pointer table.

**(2) FLXAnFRITC.IQEIE**

Input Queue Empty Interrupt Enable Bit

This bit controls the input queue empty interrupt.

0: Disabled
No interrupt will be requested and the input queue empty interrupt line will be released.

1: Enabled
Input queue empty interrupt will be asserted when FLXAnFRITS.IQEIS is 1.

**(3) FLXAnFRITC.IQFIE**

Input Queue Full Interrupt Enable Bit

This bit controls the input queue full interrupt.

0: Disabled
No interrupt will be requested and the input queue full interrupt line will be released.

1: Enabled
Input queue full interrupt will be asserted when FLXAnFRITS.IQFIS is 1.

**(4) FLXAnFRITC.IQHR**

Input Queue Halt Request Bit

The IQHR bit should not be set to 1 when FLXAnFRITS.ITS is 0.

This bit requests a halt of the input queue.

The status of the halt request is shown in the FLXAnFRITS.IQH register.

See **Section 20.3.16.1 (5), Halting the input queue** about usage of this bit.

0: Input queue run request
The input queue resumes their operation.

1: Input queue halt request
The input queue gets halted. An active input transfer will be completed but no further transfer request will start.

**(5) FLXAnFRITC.ITE**

Input Transfer Enable Bit

The user should only set this bit to 1 when FLXAnFRIBCR.IBSYS is 0.

The user should only set this bit to 0 when FLXAnFRITC.IQHR 0. Otherwise committed input transfers get lost.

This bit controls the operation mode of the input transfer queue.

The operation status of the input transfer queue function is shown in FLXAnFRITS.ITS.

See **Section 20.3.16.1 (1), Activation and deactivation** about usage of this bit.

0: Operation Disable request
The input transfer queue gets disabled when it becomes empty.

1: Operation Enable request
The input transfer queue gets enabled. Input data structures are transferred to the FlexRay internal message RAM.

### 20.2.13.2 FLXAnFROTC — FlexRay Output Transfer Configuration Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0804_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | FTM[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | FWIE | OWIE | FIE | OIE | — | — | — | — | — | — | OTCS | OTE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

**Table 20.82 FLXAnFROTC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 20 to 16 | FTM[4:0] | FIFO Table Max Bit<br>Configures the number of FIFO entries the output transfer handler is capable to maintain in the Local RAM/Global RAM. |
| 15 to 12 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 11 | FWIE | FIFO transfer Warning Interrupt Enable Bit<br>0: Disabled<br>1: Enabled |
| 10 | OWIE | Output transfer Warning Interrupt Enable Bit<br>0: Disabled<br>1: Enabled |
| 9 | FIE | FIFO transfer Interrupt Enable Bit<br>0: Disabled<br>1: Enabled |
| 8 | OIE | Output transfer Interrupt Enable Bit<br>0: Disabled<br>1: Enabled |
| 7 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | OTCS | Output Transfer Condition Select Bit<br>0: New data only mode<br>1: New data and status changed mode |
| 0 | OTE | Output Transfer Enable Bit<br>0: Operation Disable request<br>1: Operation Enable request |

**(1) FLXAnFROTC.FTM**

FIFO Table Max Bit

The user can only write to these bits when FLXAnFROTS.OTS is 0.

Configures the number of FIFO entries the output transfer handler is capable to maintain in the Local RAM/Global RAM.

Valid values are $00_H$ (1 FIFO entry) to $1F_H$ (32 FIFO entries).

**(2) FLXAnFROTC.FWIE**

FIFO transfer Warning Interrupt Enable Bit

This bit controls the FIFO transfer warning interrupt.

0: Disabled
No interrupt will be requested and the FIFO transfer warning interrupt line will be released.

1: Enabled
FIFO transfer warning interrupt will be asserted when FLXAnFROTS.FWIS is 1.

**(3) FLXAnFROTC.OWIE**

Output transfer Warning Interrupt Enable Bit

This bit controls the output transfer warning interrupt.

0: Disabled
No interrupt will be requested and the output transfer warning interrupt line will be released.

1: Enabled
Output transfer warning interrupt will be asserted when FLXAnFROTS.OWIS is 1.

**(4) FLXAnFROTC.FIE**

FIFO transfer Interrupt Enable Bit

This bit controls the FIFO transfer interrupt.

0: Disabled
No interrupt will be requested and the FIFO transfer interrupt line will be released.

1: Enabled
FIFO transfer interrupt will be asserted when FLXAnFROTS.FIS is 1.

**(5) FLXAnFROTC.OIE**

Output transfer Interrupt Enable Bit

This bit controls the output transfer interrupt.

0: Disabled
No interrupt will be requested and the output transfer interrupt line will be released.

1: Enabled
Output transfer interrupt will be asserted when FLXAnFROTS.OTIS is 1.

### (6) FLXAnFROTC.OTCS

Output Transfer Condition Select Bit

The user can only write to this bit when FLXAnFROTS.OTS is 0.

This bit controls the output transfer condition.

0: New data only mode
The ND bits in the FLXAnFRNDATi registers are used to detect a transfer condition for dedicated receive buffer

1: New data and status changed mode
The ND bits in the FLXAnFRNDATi registers and the MBSC bits in the FLXAnFRMBSC register are used to detect a transfer condition for dedicated transmit and receive buffer

### (7) FLXAnFROTC.OTE

Output Transfer Enable Bit

The user should only set this bit to 1 when FLXAnFROBCR.OBSYS is 0.

This bit controls the operation mode of the output transfer function.

The operation status of the output buffer transfer function is shown in FLXAnFROTS.OTS.

See **Section 20.3.16.2 (2), Output transfer data structure** about usage of this bit.

0: Operation Disable request
The output buffer transfer gets disabled.
An active message buffer transfer will be completed but no further transfer will start.

1: Operation Enable request
The output buffer transfer gets enabled. Message buffers are transferred from the FlexRay internal message RAM to output data structures.

The user should not change the E-Ray message RAM configuration by writing to the FLXAnFRMRC register.

### 20.2.13.3 FLXAnFRIBA — FlexRay Input Pointer Table Base Address Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0808_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ITA[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ITA[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

**Table 20.83 FLXAnFRIBA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ITA[31:0] | Input Table Address Bit<br>These bits configure the base address of the input pointer table. |

#### (1) FLXAnFRIBA.ITA

Input Table Address Bit

The user can only write to this bit when FLXAnFRITS.ITS is 0.

The address should be 32 bit aligned, thus the bits FLXAnFRIBA.ITA[1:0] are always 0.

These bits configure the base address of the input pointer table.

The table is used for the input transfer queue transferring message buffers from the Local RAM/Global RAM into the FlexRay internal message RAM.

The size of the input queue is configured in FLXAnFRITC.ITM.

Note that each entry requires two long words in the input pointer table.

### 20.2.13.4 FLXAnFRFBA — FlexRay FIFO Pointer Table Base Address Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $080C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FTA[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FTA[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

**Table 20.84 FLXAnFRFBA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | FTA[31:0] | FIFO pointer Table Address Bit<br>These bits configure the base address of the FIFO pointer table. |

#### (1) FLXAnFRFBA.FTA

FIFO pointer Table Address Bit

The user can only write to this bit when FLXAnFROTS.OTS is 0.

The address should be 32 bit aligned, thus the bits FLXAnFRFBA.FTA[1:0] are always 0.

These bits configure the base address of the FIFO pointer table.

The table is used for message buffers transferred from the FlexRay internal FIFO to the Local RAM/ Global RAM.

The size of the FIFO is configured in FLXAnFROTC.FTM.

### 20.2.13.5 FLXAnFROBA — FlexRay Output Pointer Table Base Address Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0810_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OTA[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OTA[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

**Table 20.85 FLXAnFROBA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | OTA[31:0] | Output pointer Table Address Bit<br>These bits configure the base address of the output pointer table. |

#### (1) FLXAnFROBA.OTA

Output pointer Table Address Bit

The user can only write to this bit when FLXAnFROTS.OTS is 0.

The address should be 32 bit aligned, thus the bits FLXAnFROBA.OTA[1:0] are always 0.

These bits configure the base address of the output pointer table.

The table is used for message buffers transferred from the FlexRay internal message RAM to the Local RAM/Global RAM.

The size of the table depends on the utilization of the FlexRay internal message RAM and can have up to 128 entries.

### 20.2.13.6 FLXAnFRIQC — FlexRay Input Queue Control Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0814_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | IMBNR[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W |

**Table 20.86 FLXAnFRIQC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 6 to 0 | IMBNR[6:0] | Input Message Buffer NumbeR Bit<br>Message buffer number added to the input queue |

#### (1) FLXAnFRIQC.IMBNR

Input Message Buffer NumbeR Bit

The user can only write to this bit when FLXAnFRITS.IQFP is 0.

The user should not write to this register when FLXAnFRITS.ITS is 0 or when FLXAnFRITC.ITE is 0.

These bits are read as 0.

This value specifies the message buffer added to the input queue.

The number has to be identical to FLXAnFRWRHS4.IMBNR (see **Section 20.3.16.1 (3), Input pointer table**) of the input pointer table.

The address to the input data structure has to be provided in the input pointer table at the put index (FLXAnFRITS.IPIDX) before writing to this register.

Writing to this register increments the input put index (FLXAnFRITS.IPIDX).

### 20.2.13.7 FLXAnFRUIR — FlexRay User Input Transfer Request Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0818_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | UIDX[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.87 FLXAnFRUIR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 7 to 0 | UIDX[7:0] | User requested Input inDeX Bit<br>Input pointer table index requested for input transfer |

#### (1) FLXAnFRUIR.UIDX

User requested Input inDeX Bit

The user can only write to this bit when FLXAnFRITS.UIRP is 0.

The user should not write to this register when FLXAnFRITS.ITS is 0.

The user should not write to this register when FLXAnFRITS.UIRP is 1.

The user should not write to this register when FLXAnFRITS.IQH is 1.

The user should only write FLXAnFRITC.ITM + 1 to this register.

This value specifies the input pointer table index for the user requested input transfer.

The address to the input data structure has to be provided in the input pointer table at the index UIDX before writing to this register.

When writing to this register, the requested input data structure will be transferred from input data structure position to the FlexRay internal message RAM.

In opposite to queued input transfers the related DA flag in the FLXAnFRDA register is not influenced by the user input transfer.

### 20.2.13.8 FLXAnFRUOR — FlexRay User Output Transfer Request Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $081C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | URDS | — | — | UMBNR[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.88 FLXAnFRUOR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 9 | URDS | User request Read Data Section Bit<br>0: Data section is not transferred<br>1: Data section is transferred |
| 8, 7 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 6 to 0 | UMBNR[6:0] | User requested output Message Buffer NumbeR Bit<br>Message buffer number requested for output transfer |

#### (1) FLXAnFRUOR.URDS

User request Read Data Section Bit

The user can only write to this bit when FLXAnFROTS.UORP is 0.

The user should not write to this register when FLXAnFROTS.OTS is 0.

The user should not write to this register when FLXAnFROTS.UORP is 1.

0: Data section is not transferred
The data section of the message buffer selected by the bits UMBNR is not requested

1: Data section is transferred
The data section of the message buffer selected by the bits UMBNR is requested

#### (2) FLXAnFRUOR.UMBNR

User requested output Message Buffer NumbeR Bit

The user can only write to this bit when FLXAnFROTS.UORP is 0.

The user should not write to this register when FLXAnFROTS.OTS is 0.

The user should not write to this register when FLXAnFROTS.UORP is 1.

The user should restrict UMBNR to dedicated receive and transmit buffers when the FlexRay module is not in the CONFIG state.

When writing to this register, the header sections and optionally the data section (configurable by URDS) of the requested message buffer will be transferred from the FlexRay internal message RAM to the output data structure position defined by the output structure data pointer in the output pointer table.

## 20.2.14 Data Transfer Status Register

### 20.2.14.1 FLXAnFRITS — FlexRay Input Transfer Status Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0820_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | IGIDX[6:0] | | | | | | | — | IPIDX[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | IQFP | — | — | IQEIS | IQFIS | — | — | — | — | — | UIRP | IQH | ITS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R | R |

**Table 20.89 FLXAnFRITS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 30 to 24 | IGIDX[6:0] | Input queue Get InDeX Bit<br>Represents the get index of the input pointer table |
| 23 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 22 to 16 | IPIDX[6:0] | Input queue Put InDeX Bit<br>Represents the put index of the input pointer table |
| 15 to 13 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 12 | IQFP | Input Queue Full condition Pending Bit<br>0: Entries in the input queue are available<br>1: All entries in the input queue are occupied |
| 11, 10 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 9 | IQEIS | Input Queue Empty Interrupt Status Bit<br>0: No input queue empty condition detected<br>1: Input queue empty condition detected |
| 8 | IQFIS | Input Queue Full Interrupt Status Bit<br>0: No input queue full condition detected<br>1: Input queue full condition detected |
| 7 to 3 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 2 | UIRP | User Input transfer Request Pending Bit<br>0: No user input transfer request pending<br>1: User input transfer request pending |
| 1 | IQH | Input Queue Halted Bit<br>0: Input queue not halted<br>1: Input queue halted |
| 0 | ITS | Input Transfer Status Bit<br>0: Disabled<br>1: Enabled |

**(1) FLXAnFRITS.IGIDX**

Input queue Get InDeX Bit

These bits are only valid when FLXAnFRITS.IQH is 1

These bits represent the input pointer index the input queue handler will transfer next.

Valid values are $00_H$ to FLXAnFRITC.ITM.

The get index is incremented when the input data structure has been transferred from the Local RAM/ Global RAM and the related DA flag in the FLXAnFRDA register is cleared.

The index is set to $00_H$ when FLXAnFRITS.ITS changes from 0 to 1.

**(2) FLXAnFRITS.IPIDX**

Input queue Put InDeX Bit

These bits represent the index where the next input data structure pointer in the input pointer table should be stored.

Valid values are $00_H$ to FLXAnFRITC.ITM.

After reaching the maximum value the put index continues from $00_H$.

The index is incremented when writing to FLXAnFRIQC.IMBNR.

The index is set to $00_H$ when FLXAnFRITS.ITS changes from 0 to 1.

**(3) FLXAnFRITS.IQFP**

Input Queue Full condition Pending Bit

This bit represents that the input queue is full.

There should be no further input transfer requests, by writing to FLXAnFRIQC.IMBNR, as long as FLXAnFRITS.IQFP is 1.

[Clearing condition]

This bit is cleared when there is one free entry in the input queue.

[Setting condition]

This bit is set when all entries in the input queue are occupied.

**(4) FLXAnFRITS.IQEIS**

Input Queue Empty Interrupt Status Bit

Writing 0 has no effect on the bit value.

If enabled in FLXAnFRITC.IQEIE the input queue empty interrupt is generated when FLXAnFRITS.IQEIS is 1.

[Clearing condition]

This bit is cleared when writing a 1 to FLXAnFRITS.IQEIS.
This bit is cleared when FLXAnFRITS.ITS changes from 0 to 1.

[Setting condition]

This bit is set when all pending input transfers have been processed and consequently the input queue becomes empty.

**(5) FLXAnFRITS.IQFIS**

Input Queue Full Interrupt Status Bit

Writing 0 has no effect on the bit value.

If enabled in FLXAnFRITC.IQFIE the input queue full interrupt is generated when FLXAnFRITS.IQFIS is 1.

This flag is intended as interrupt status flag. It does not represent the current input queue status; for this status see FLXAnFRITS.IQFP.

[Clearing condition]

This bit is cleared when writing a 1 to FLXAnFRITS.IQFIS.

This bit is cleared when FLXAnFRITS.ITS changes from 0 to 1.

[Setting condition]

This bit is set when all entries in the input queue are occupied.

**(6) FLXAnFRITS.UIRP**

User Input transfer Request Pending Bit

This bit represents that a user input transfer is still pending.

There should be no further write access to FLXAnFRUIR.UIDX when this bit is 1.

[Clearing condition]

This bit is cleared when the user input transfer request is processed by the input transfer handler.

[Setting condition]

This bit is set when writing to FLXAnFRUIR.UIDX.

**(7) FLXAnFRITS.IQH**

Input Queue Halted Bit

This bit represents the status of the input queue.

There should be no further write access to FLXAnFRUIR.UIDX when this bit is 1.

[Clearing condition]

This bit is cleared when FLXAnFRITC.IQHR is set to 0.

[Setting condition]

This bit is set immediately when the FLXAnFRITC.IQHR is set to 1 and there is no ongoing input transfer.
This bit is set only after an ongoing input transfer has been completed and FLXAnFRITC.IQHR is set to 1.

**(8) FLXAnFRITS.ITS**

Input Transfer Status Bit

This bit represents the status of the input queue handler.

While this bit is 1, there should be no read or write access to the address area <FLXn_base> + $0400_H$ to <FLXn_base> + $05FF_H$ and there should be no CLEAR_RAMS command applied to FLXAnFRSUCC1.CMD register.

The input transfer queue indexes and related status flags are set to 0 when FLXAnFRITS.ITS changes from 0 to 1.

[Clearing condition]

This bit is cleared immediately when FLXAnFRITC.ITE is set to 0 and there are no pending input transfers

This bit is cleared after all pending requests have been processed and FLXAnFRITC.ITE is 0.

[Setting condition]

This bit is set when FLXAnFRITC.ITE is set to 1.

### 20.2.14.2 FLXAnFROTS — FlexRay Output Transfer Status Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0824_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | FFL[5:0] | | | | | | — | — | — | FGIDX[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FWP | OWP | FDA | — | FWS | OWS | FIS | OTIS | — | — | — | — | — | UORP | — | OTS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

**Table 20.90 FLXAnFROTS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 29 to 24 | FFL[5:0] | FIFO Fill Level Bit<br>Represent the number of unprocessed output FIFO structures |
| 23 to 21 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 20 to 16 | FGIDX[4:0] | FIFO Get InDeX Bit<br>Represent the get index in the FIFO pointer table |
| 15 | FWP | FIFO transfer Warning condition Pending Bit<br>0: No FIFO transfer warning condition pending<br>1: FIFO transfer warning condition pending |
| 14 | OWP | Output transfer Warning condition Pending<br>0: No output transfer warning condition pending<br>1: Output transfer warning condition pending |
| 13 | FDA | FIFO Data Available Bit<br>0: No available FIFO structures<br>1: FIFO structures available at current FLXAnFROTS.FGIDX index |
| 12 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 11 | FWIS | FIFO transfer Warning Interrupt Status Bit<br>0: No FIFO transfer warning condition detected<br>1: FIFO transfer warning condition detected |
| 10 | OWIS | Output transfer Warning Interrupt Status Bit<br>0: No output transfer warning condition detected<br>1: Output transfer warning condition detected |
| 9 | FIS | FIFO transfer Interrupt Status Bit<br>0: No FIFO structure updated in Local RAM/Global RAM<br>1: FIFO structure updated in Local RAM/Global RAM |
| 8 | OTIS | Output transfer Interrupt Status Bit<br>0: No output structure updated in Local RAM/Global RAM<br>1: Output structure updated in Local RAM/Global RAM |
| 7 to 3 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |

Table 20.90 FLXAnFROTS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | UORP | User Output transfer Request Pending Bit<br>0: No user output transfer request pending<br>1: User output transfer request pending |
| 1 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 0 | OTS | Output Transfer Status Bit<br>0: Disabled<br>1: Enabled |

### (1) FLXAnFROTS.FFL

FIFO Fill Level Bit

These bits represent the number of available output FIFO structures in the Local RAM/Global RAM.

Valid values are $00_H$ to FLXAnFROTC.FTM + 1.

The value $00_H$ represents that the FIFO is empty.

The value FLXAnFROTC.FTM + 1 represents that the FIFO is full and no further FIFO transfers will be done.

The FIFO fill level is incremented when a FIFO data structure has been transferred from the FlexRay internal FIFO into the Local RAM/Global RAM.

The FIFO fill level is decremented when the user releases a FIFO data structure in the Local RAM/Global RAM by writing 1 to FLXAnFROTS.FDA.

The FIFO fill level is set to $00_H$ when the bit FLXAnFROTS.OTS changes from 0 to 1.

### (2) FLXAnFROTS.FGIDX

FIFO Get InDeX Bit

These bits represent the index where the current output data structure pointer in the FIFO pointer table is available for reading.

Valid values are $00_H$ to FLXAnFROTC.FTM.

After reaching the maximum value the get index continues from $00_H$.

The index is incremented when a FIFO data structure is released by writing 1 to FLXAnFROTS.FDA.

The index is set to $00_H$ when FLXAnFROTS.OTS changes from 0 to 1.

### (3) FLXAnFROTS.FWP

FIFO transfer Warning condition Pending Bit

This bit represents the FIFO transfer warning condition.

[Clearing condition]

This bit is cleared when there are free output data structures (FLXAnFROTS.FFL <= FLXAnFROTC.FTM).

This bit is cleared when the FLXAnFROTS.OTS changes from 0 to 1.

[Setting condition]

This bit is set when the output transfer handler detects a transfer condition for FIFO message buffers but there are no free output data structures (FLXAnFROTS.FFL = FLXAnFROTC.FTM + 1).

### (4) FLXAnFROTS.OWP

Output transfer Warning condition Pending Bit

This bit represents the output transfer warning condition.

[Clearing condition]

This bit is cleared, when all output structure pointers that have a pending output handler transfer condition detected, are released (for dedicated transmit and receive message buffers or a user output transfer request).

[Setting condition]

This bit is set when the output transfer handler detects a transfer condition (for dedicated transmit and receive message buffers or a user output transfer request) but the related output structure pointer was not yet released by the application (data available flag is still set to 1).

This bit is set when the output transfer handler detects a transfer condition for dedicated transmit and receive message buffers but there is a pending input transfer for the same message buffer (data available flag is set to 1 due to the input transfer request).

### (5) FLXAnFROTS.FDA

FIFO Data Available Bit

Writing 0 has no effect on the bit value.

When this bit is 1, the next valid output data structure is available.

The related data structure pointer is in the FIFO pointer table at FLXAnFROTS.FGIDX.

Writing 1 to FLXAnFROTS.FDA

- increments FLXAnFROTS.FGIDX and
- decrements the FIFO fill level (FLXAnFROTS.FFL)

If there are still unprocessed data structures FLXAnFROTS.FDA remains 1.

[Clearing condition]

This bit is cleared when writing 1 to FLXAnFROTS.FDA and the FIFO fill level becomes $00_H$.

This bit is cleared when the FLXAnFROTS.OTS changes from 0 to 1.

[Setting condition]

This bit is set when there is at least one FIFO data structure available in the Local RAM/Global RAM.

### (6) FLXAnFROTS.FWIS

FIFO transfer Warning Interrupt Status Bit

Writing 0 has no effect on the bit value.

If enabled in FLXAnFROTC.FWIE the FIFO transfer warning interrupt is generated when FLXAnFROTS.FWIS is 1.

[Clearing condition]

This bit is cleared when writing a 1 to FLXAnFROTS.FWIS.

This bit is cleared when the bit FLXAnFROTS.OTS changes from 0 to 1.

[Setting condition]

This bit is set when the output transfer handler detects a transfer condition for FIFO message buffers but there are no free output data structures (FLXAnFROTS.FFL = FLXAnFROTC.FTM + 1).

### (7) FLXAnFROTS.OWIS

Output transfer Warning Interrupt Status Bit

Writing 0 has no effect on the bit value.

If enabled in FLXAnFROTC.OWIE the FIFO transfer warning interrupt is generated when FLXAnFROTS.OWIS is 1.

[Clearing condition]

This bit is cleared when writing a 1 to FLXAnFROTS.OWIS.

This bit is cleared when the bit FLXAnFROTS.OTS changes from 0 to 1.

[Setting condition]

This bit is set when the output transfer handler detects a transfer condition (for dedicated transmit and receive message buffers or a user output transfer request) but the related output structure pointer was not yet released by the application (data available flag is still set to 1).

This bit is set when the output transfer handler detects a transfer condition for dedicated transmit and receive message buffers but there is a pending input transfer for the same message buffer (data available flag is set to 1 due to the input transfer request).

### (8) FLXAnFROTS.FIS

FIFO transfer Interrupt Status Bit

Writing 0 has no effect on the bit value.

If enabled in FLXAnFROTC.FIE the FIFO transfer interrupt is generated when FLXAnFROTS.FIS is 1.

[Clearing condition]

This bit is cleared when writing a 1 to FLXAnFROTS.FIS.

This bit is cleared when the bit FLXAnFROTS.OTS changes from 0 to 1.

[Setting condition]

This bit is set when a FIFO data structure is updated by the transfer handler or the FFL bit changes from $00_H$ to $01_H$.

### (9) FLXAnFROTS.OTIS

Output transfer Interrupt Status Bit

Writing 0 has no effect on the bit value.

If enabled in FLXAnFROTC.OIE the output transfer interrupt is generated when FLXAnFROTS.OTIS is 1.

[Clearing condition]

This bit is cleared when writing a 1 to FLXAnFROTS.OTIS.
This bit is cleared when the bit FLXAnFROTS.OTS changes from 0 to 1.

[Setting condition]

This bit is set when an output data structure is updated by the transfer handler (from a dedicated transmit or receive message buffer or by a user output transfer request).

### (10) FLXAnFROTS.UORP

User Output transfer Request Pending Bit

This bit represents that a user output transfer is still pending.

There should be no further write access to FLXAnFRUOR.UMBNR when this bit is 1.

[Clearing condition]

This bit is cleared when the user output transfer request is processed by the output transfer handler.

This bit is cleared when the bit FLXAnFROTS.OTS changes from 0 to 1.

[Setting condition]

This bit is set when writing to FLXAnFRUOR.UMBNR.

### (11) FLXAnFROTS.OTS

Output Transfer Status Bit

This bit represents the status of the output transfer handler.

While this bit is 1, there should be no read or write access to the address area <FLXn_base> + $0600_H$ to <FLXn_base> + $07FF_H$ and there should be no CLEAR_RAMS command applied to FLXAnFRSUCC1.CMD register.

While this bit is 1, the user should not change the E-Ray message RAM configuration by writing to the FLXAnFRMRC register.

The output hander transfer indexes and related status flags are set to 0 when FLXAnFROTS.OTS changes from 0 to 1.

[Clearing condition]

This bit is cleared immediately when FLXAnFROTC.OTE is set to 0 and there are no ongoing output transfers.

This bit is cleared after an ongoing transfer has been completed and FLXAnFROTC.OTE is 0.

[Setting condition]

This bit is set when FLXAnFROTC.OTE is set to 1.

### 20.2.14.3 FLXAnFRAES — FlexRay Access Error Status Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0828_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MAE | FAE | OAE | IAE | EIDX[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

**Table 20.91 FLXAnFRAES Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 12 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 11 | MAE | Multiple Access Errors Bit<br>0: No multiple access errors occurred<br>1: Multiple access errors occurred |
| 10 | FAE | FIFO transfer Access Error Bit<br>0: No access error occurred during FIFO transfer<br>1: Access error occurred during FIFO transfer |
| 9 | OAE | Output transfer Access Error Bit<br>0: No access error occurred during output transfer<br>1: Access error occurred during output transfer |
| 8 | IAE | Input transfer Access Error Bit<br>0: No access error occurred during input transfer<br>1: Access error occurred during input transfer |
| 7 to 0 | EIDX[7:0] | Error InDeX Bit<br>Data structure pointer index number |

**(1) FLXAnFRAES.MAE**

Multiple Access Errors Bit

Writing 0 has no effect on the bit value.

This bit represents that there were multiple access errors during a data transfer.

[Clearing condition]

This bit is cleared when writing a 1 to FLXAnFRAES.MAE.

[Setting condition]

This bit is set when one of the bits FLXAnFRAES.FAE, FLXAnFRAES.OAE or FLXAnFRAES.IAE are set and

- an access to a protected address occurred during a FIFO data transfer or
- an access to a protected address occurred during an output data transfer or
- an access to a protected address occurred during an input data transfer

**(2) FLXAnFRAES.FAE**

FIFO transfer Access Error Bit

Writing 0 has no effect on the bit value.

This bit represents that there was an access error during a FIFO data transfer.

[Clearing condition]

This bit is cleared when writing a 1 to FLXAnFRAES.FAE.

[Setting condition]

This bit is set when a Local RAM/Global RAM access error was detected during a FIFO transfer and the bits FLXAnFRAES.OAE, FLXAnFRAES.IAE and FLXAnFRAES.MAE are 0.

**(3) FLXAnFRAES.OAE**

Output transfer Access Error Bit

Writing 0 has no effect on the bit value.

This bit represents that there was an access error during an output data transfer.

[Clearing condition]

This bit is cleared when writing a 1 to FLXAnFRAES.OAE.

[Setting condition]

This bit is set when a Local RAM/Global RAM access error was detected during an output transfer and the bits FLXAnFRAES.FAE, FLXAnFRAES.IAE and FLXAnFRAES.MAE are 0.

**(4) FLXAnFRAES.IAE**

Input transfer Access Error Bit

Writing 0 has no effect on the bit value.

This bit represents that there was an access error during an input data transfer.

[Clearing condition]

This bit is cleared when writing a 1 to FLXAnFRAES.IAE.

[Setting condition]

This bit is set when a Local RAM/Global RAM access error was detected during an output transfer and the bits FLXAnFRAES.OAE, FLXAnFRAES.FAE and FLXAnFRAES.MAE are 0.

### (5) FLXAnFRAES.EIDX

Error InDeX Bit

This value is only valid when one of the bits FLXAnFRAES.FAE, FLXAnFRAES.OAE or FLXAnFRAES.IAE is 1.

When the bit FLXAnFRAES.FAE is 1, FLXAnFRAES.EIDX holds the used FIFO put index when the access error has occurred.

When the bit FLXAnFRAES.OAE is '1', FLXAnFRAES.EIDX holds the used output table entry (which is related to message buffer number) when the access error has occurred.

When the bit FLXAnFRAES.IAE is '1', FLXAnFRAES.EIDX holds the used input pointer table get index when the access error has occurred during the input transfer or user requested input transfer.

These bits are updated when one of the bits FLXAnFRAES.FAE, FLXAnFRAES.OAE or FLXAnFRAES.IAE is changing from 0 to 1.

### 20.2.14.4 FLXAnFRAEA — FlexRay Access Error Address Register

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $082C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AEA[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AEA[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.92 FLXAnFRAEA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | AEA[31:0] | Access Error Address Bit<br>Address in the Local RAM/Global RAM when an access error has occurred |

#### (1) FLXAnFRAEA.AEA

Access Error Address Bit

This value is only valid when one of the bits FLXAnFRAES.FAE, FLXAnFRAES.OAE or FLXAnFRAES.IAE is 1.

These bits represent the address of the access error indicated in the FLXAnFRAES register.

These bits are updated when one of the bits FLXAnFRAES.FAE, FLXAnFRAES.OAE or FLXAnFRAES.IAE is changing from 0 to 1.

### 20.2.14.5 FLXAnFRDAi — FlexRay message Data Available Register i (i = 0 to 3)

**Access:** This register can be read/written in 8-, 16-, or 32-bit units.

**Address:** <FLXn_base> + $0830_H$ to <FLXn_base> + $083C_H$ (<FLXn_base> + $0830_H$ + i*4)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DAb[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DAb[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.93 FLXAnFRDAi Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DAb[31:0] | Data Available Bit<br>0: No data available for destination<br>1: Data available for destination |

#### (1) FLXAnFRDAi.DAb (b = i*32 to ((i+1)*32) –1)

Data Available Bit b

The user should not write a 1 to bits that are 0.

To maintain the status of input transfers, the user should not clear bits related to input transfers.

This register is used for input and output transfers.

Each flag corresponds to a FlexRay message buffer.

[Clearing condition]

Input transfer:
This bit is cleared when the input data structure has been transferred from the Local RAM/Global RAM. The data structure and the data structure pointer can be changed when the related flag is 0.

Output transfer:
This bit is cleared when writing a 1 to it.

[Setting condition]

Input transfer:
This bit is set when the corresponding message buffer number has been written to FLXAnFRIQC.IMBNR.

As long as this bit is 1, the input data structure and the data structure pointer corresponding to this input transfer request should not be changed.

Output transfer:

This bit is set when the output data structure corresponding to this message buffer has been updated. As long as this bit is 1, the data structure is stable; no further update of the data structure will be done by the output handler. While this bit is 1, the application is allowed to change the output data structure pointer in the output pointer table for this message buffer number.

## 20.2.15 NTU loss monitor register

### 20.2.15.1 NTU0RST — NTU loss monitor reset register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF9 CC00$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | NTUCRST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 20.94 NTU0RST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 0 | NTUCRST | NTU reset<br>0: No function<br>1: Reset the NTU operation |

#### (1) NTU0RST.NTUCRST

NTU loss monitor reset

Setting this bit resets the NTU operation. This bit always returns 0 when it is read.

### 20.2.15.2 NTU0CFG — NTU configuration register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF9 CC04$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CLKSEL | — | — | — | — | — | NTUSEL[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R/W | R/W |

**Table 20.95 NTU0CFG Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 7 | CLKSEL | NTU Output Clock Select<br>0: NTU signal is output<br>1: Backup clock is output |
| 6 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1, 0 | NTUSEL[1:0] | NTU Input Signal Select<br>00: FLXA0 macro tick signal is input to NTU (MT0)<br>01: FLXA0 cycle tick signal is input to NTU (CS0)<br>10: FLXA1 macro tick signal is input to NTU (MT1)<br>11: FLXA1 cycle tick signal is input to NTU (CS1) |

#### (1) NTU0CFG.CLKSEL

NTU output clock select.

This bit is used to select between NTU clock signal and Backup clock signal as the output of NTU. After detection of the NTU loss signal, the application has to switch the NTU output clock using this bit.

#### (2) NTU0CFG.NTUSEL

NTU input clock select.

This bit is used to select between macro tick clock signal and cycle tick clock signal as the monitored input of NTU. Additionally it can be selected if FlexRay macro 0 or 1 should be used as a source.

**NOTE**

Not all devices are equipped with two FlexRay macros. In case only one macro is available, please do not set the NTUSEL[1:0] bits to “10” or “11”.

### 20.2.15.3 NTU0CLMEN — NTU enable register

**Access:** This register can be read/written in 32-bit units.

**Address:** FFF9 CC08$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | NTUCEN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 20.96 NTU0CLMEN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 0 | NTUCEN | NTU Enable<br>0: Disable NTU operation<br>1: Enable NTU operation |

#### (1) NTU0CLMEN.NTUCEN

NTU enable bit.

This bit is used to enable or disable NTU operation.

### 20.2.15.4 NTU0MON — NTU status register

**Access:** This register can be read in 32-bit units.

**Address:** FFF9 CC0C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | NTUMON | NTUCMON |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 20.97 NTU0MON Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1 | NTUMON | NTU Monitor Status<br>0: Monitored NTU input is lost<br>1: Monitored NTU input is correct |
| 0 | NTUCMON | NTU Status<br>0: NTU does not operate<br>1: NTU is operating |

#### (1) NTU0MON.NTUMON

NTU monitor status bit.

This bit shows the status of the monitored input signal.

#### (2) NTU0MON.NTUCMON

NTU status bit.

This bit shows the status of the NTU operation.

### 20.2.15.5 NTU0PRSCTL — NTU prescaler control register

**Access:** This register can be read/written in 8-bit units.

**Address:** FFF9 C008$_H$

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | NPRSCE | — | — | NPRSCCS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R | R | R/W | R/W |

**Table 20.98 NTU0PRSCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 4 | NPRSCE | NTU Prescaler Enable<br>0: Disables NTU prescaler<br>1: Enables NTU prescaler |
| 3, 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 0 | NPRSCCS[1:0] | NTU Prescaler Selection<br>00: Prescaler counter clock is CLKP_H1<br>01: Prescaler counter clock is CLKP_H1/2<br>10: Prescaler counter clock is CLKP_H1/4<br>11: Prescaler counter clock is CLKP_H1/8 |

#### (1) NTU0NPRSCTL.NPRSCE

NTU prescaler enable bit.

This bit enables the NTU prescaler (NPRS). See **Figure 20.33**.

#### (2) NTU0NPRSCTL.NPRSCCS

NTU prescaler selection bits.

These bits set the scaling factor of the input clock (CLKP_H1).

### 20.2.15.6 NTU0PRSCMP — NTU prescaler compare register

**Access:** This register can be read/written in 8-bit units.

**Address:** FFF9 C00C$_H$

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NPRSCMP[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.99 NTU0PRSCMP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | NPRSCMP[7:0] | NTU Prescaler Compare Value<br>Compare match value of the NTU prescaler counter |

#### (1) NTU0PRSCMP.NPRSCMP

NTU prescaler compare match value bits.

These bits set the NTU prescaler output clock period according to the equation below:

Prescaler_output_clock_period = NPRS_counter_clock / (2 * NPRSCMP)

NPRS_counter_clock is selected using the NTU0PRSCTL register.

### 20.2.15.7 NTU0CCTL0 — NTU compare unit control register

**Access:** This register can be read/written in 8-bit units.

**Address:** FFF9 C800$_H$

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | NTUCCMPE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 20.100 NTU0CCTL0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 0 | NTUCCMPE | NTU Compare Unit Enable<br>0: Disable NTU compare unit operation<br>1: Enable NTU compare unit operation |

#### (1) NTU0CCTL0.NTUCCMPE

NTU compare unit enable bit.

This bit is used to enable or disable NTU compare unit operation.

### 20.2.15.8 NTU0CCMPH — NTU compare unit threshold register H

**Access:** This register can be read/written in 16-bit units.

**Address:** FFF9 C80C$_H$

**Value after reset:** 03FF$_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | NTUCCMPH[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.101 NTU0CCMPH Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 11 to 0 | NTUCCMPH [11:0] | Compare Unit Upper Threshold<br>Sets the upper threshold value of the compare unit |

#### (1) NTU0CCMPH.NTUCCMPH[11:0]

NTU compare unit upper threshold value.

The value set in these bits is used as the upper threshold value for the compare unit. The compare unit counter is counting up using the Base clock (see **Figure 20.33**). Additionally the NTU ticks are counted. If the amount of the NTU ticks fits into the window set by the NTU0CCMPH and NTU0CCMPL registers, the NTU loss is not detected. If the amount of NTU ticks is lower than NTUCCMPL register value, or higher than NTU0CCMPH register value the NTU loss is detected and it will be signalled to the application by an interrupt. The output clock signal has to be switched to the Backup clock by the application. See **Figure 20.33**.

Setting range: 004$_H$ to FFF$_H$

The setting value must be higher than NTU0CCMPL + 003$_H$.

### 20.2.15.9 NTU0CCMPL — NTU compare unit threshold register L

**Access:** This register can be read/written in 16-bit units.

**Address:** FFF9 C808$_H$

**Value after reset:** 0001$_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | NTUCCMPL[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.102 NTU0CCMPL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 12 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 11 to 0 | NTUCCMPL [11:0] | Compare Unit Upper Threshold<br>Sets the upper threshold value of the compare unit |

#### (1) NTU0CCMPL.NTUCCMPL[11:0]

NTU compare unit lower threshold value.

The value set in these bits is used as the upper threshold value for the compare unit. The compare unit counter is counting up using the Base clock (see **Figure 20.33**). Additionally the NTU ticks are counted. If the amount of the NTU ticks fits into the window set by the NTU0CCMPH and NTU0CCMPL registers, the NTU loss is not detected. If the amount of NTU ticks is lower than NTUCCMPL register value, or higher than NTU0CCMPH register value the NTU loss is detected and it will be signalled to the application by an interrupt. The output clock signal has to be switched to the Backup clock by the application. See **Figure 20.33**.

Setting range: 001$_H$ to FFC$_H$

The setting value must be higher than NTU0CCMPL + 0x003$_H$.

### 20.2.15.10NTU0CPCMD — NTU compare unit protection command register.

**Access:** This register can be written in 8-bit units.

**Address:** FFF9 C810$_H$

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NTUCREG[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W |

**Table 20.103 NTU0PCMD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | NTUCREG[7:0] | NTU Compare Unit Protection Command<br>Protection command– A5$_H$ |

#### (1) NTU0CPCMD.NTUCREG

NTU compare unit protection command bits.

Accessing NTU protected registers is allowed when protection sequence using the NTU0CPCMD register has been done. The protection sequence flow is shown below:

- Write A5$_H$ value to the NTU0CPCMD register
- Write required value to the protected register (example: 01$_H$ to NTU0CCTL0)
- Write inverted value to the protected register (FE$_H$ to NTU0CCTL0)
- Write again required value to the protected register (01$_H$ to NTU0CCTL0)

### 20.2.15.11NTU0CPS — NTU compare unit protection status register

**Access:** This register can be read in 8-bit units.

**Address:** FFF9 C814$_H$

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | NTUCPRERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 20.104 NTU0CPS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 0 | NTUCPRERR | NTU Compare Unit Protection Status<br>0: Protection error did not occur<br>1: Protection error occurred |

#### (1) NTU0CPS.NTUCPRERR

NTU compare unit protection status bits.

When a protected register has been accessed and the protection sequence has been broken, or not executed fully, an error is signalled by setting this bit to “1”.

## 20.2.15.12NTU0CPRSCTL — NTU compare unit prescaler setting register.

**Access:** This register can be read/written in 8-bit units.

**Address:** FFF9 C208$_H$

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | NCPRSCE | — | — | NCPRSCCS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R | R | R/W | R/W |

**Table 20.105 NTU0CPRSCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 4 | NCPRSCE | NTU Compare Unit Prescaler Enable<br>0: Disables NTU compare unit prescaler<br>1: Enables NTU compare unit prescaler |
| 3, 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1, 0 | NCPRSCCS [1:0] | NTU Compare Unit Prescaler Selection<br>00: Prescaler counter clock is output of NPRS<br>01: Prescaler counter clock is output of NPRS/2<br>10: Prescaler counter clock is output of NPRS/4<br>11: Prescaler counter clock is output of NPRS/8 |

### (1) NTU0CPRSCTL.NCPRSCE

NTU compare unit prescaler enable bit.

This bit enables the NTU compare unit prescaler (NCPRS). See **Figure 20.33**

### (2) NTU0CPRSCTL.NCPRSCCS

NTU compare unit prescaler selection bits.

These bits set the scaling factor of the input clock (NPRS output).

### 20.2.15.13NTU0CPRSCMP — NTU compare unit prescaler compare register

**Access:** This register can be read/written in 8-bit units.

**Address:** $FFF9C20C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NCPRSCMP[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.106 NTU0CPRSCMP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | NCPRSCMP [7:0] | NTU Compare Unit Prescaler Compare Value<br>Compare match value of the NTU compare unit counter |

#### (1) NTU0CPRSCMP.NCPRSCMP

NTU compare unit prescaler compare match value bits.

These bits set the NTU compare unit prescaler output clock period according to the equation below:

Compare_Unit_Prescaler_output_clock_period = NCPRS_counter_clock / (2 * NCPRSCMP)

NCPRS_counter_clock is selected using the NTU0CPRSCTL register.

## 20.2.15.14NTU0BPRSCTL — NTU backup clock prescaler setting register

**Access:** This register can be read/written in 8-bit units.

**Address:** FFF9 C108$_H$

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | NBPRSCE | — | — | NBPRSCCS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R | R | R/W | R/W |

**Table 20.107 NTU0BPRSCTL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 4 | NBPRSCE | NTU Backup Clock Prescaler Enable<br>0: Disables NTU backup clock prescaler<br>1: Enables NTU backup clock prescaler |
| 3, 2 | Reserved | When read, the value after reset always is read. When writing, always write the value after reset. |
| 1, 0 | NBPRSCCS[1:0] | NTU Backup Clock Prescaler Selection<br>00: Prescaler counter clock is output of NPRS<br>01: Prescaler counter clock is output of NPRS/2<br>10: Prescaler counter clock is output of NPRS/4<br>11: Prescaler counter clock is output of NPRS/8 |

### (1) NTU0BPRSCTL.NBPRSCE

NTU backup clock prescaler enable bit.

This bit enables the NTU backup clock prescaler (NBPRS). See **Figure 20.33**.

### (2) NTU0BPRSCTL.NBPRSCCS

NTU backup clock prescaler selection bits.

These bits set the scaling factor of the input clock (NPRS output).

### 20.2.15.15NTU0BPRSCMP — NTU backup clock prescaler compare register

**Access:** This register can be read/written in 8-bit units.

**Address:** FFF9 C10C$_H$

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NBPRSCMP[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 20.108 NTU0BPRSCMP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | NBPRSCMP [7:0] | NTU Backup Clock Prescaler Compare Value<br>Compare match value of the NTU backup clock counter |

#### (1) NTU0BPRSCMP.NBPRSCMP[7:0]

NTU compare unit prescaler compare match value bits.

These bits set the NTU compare unit prescaler output clock period according to the equation below:

Backup_Clock_Prescaler_output_clock_period = NBPRS_counter_clock / (2 * NBPRSCMP)

NBPRS_counter_clock is selected using the NTU0BPRSCTL register.

# 20.3 Operation

This chapter describes the FlexRay implementation together with the related FlexRay protocol features. More information about the FlexRay protocol itself can be found in the FlexRay protocol specification.

## 20.3.1 FlexRay Module Operation Control

### 20.3.1.1 FlexRay Module Enable

After hardware reset or after the FlexRay module has been disabled (following **Section 20.3.1.2, FlexRay Module Disable**) the FlexRay module is in the reset state (FLXAnFROS.OS is 0) and the clocks of the FlexRay core module are disabled.

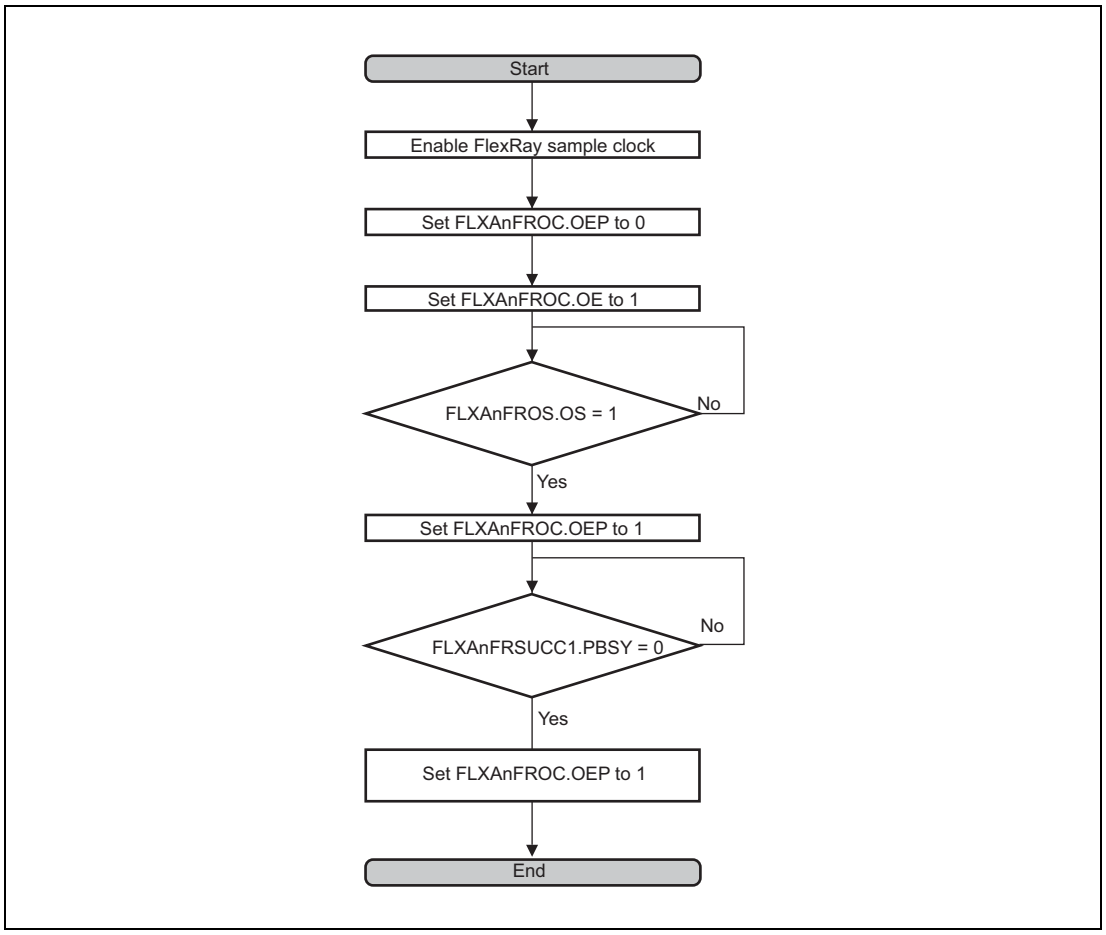



**Figure 20.3 FlexRay enable flow**

### 20.3.1.2 FlexRay Module Disable

The FlexRay module can be disabled at any time. However, it is recommended to disable the FlexRay module using the FLXAnFROC.OE register only when the FlexRay module is in HALT, CONFIG or DEFAULT_CONFIG state. Resetting the FlexRay module in any other state will terminate any ongoing FlexRay communication.

If the data transfer function is used, it is also required to disable this function before disabling the FlexRay module (see **Section 20.3.16.1 (1), Activation and deactivation** for suspending input transfer function and **Section 20.3.16.2 (2), Output transfer data structure**for suspending output transfer).

The following flow should be executed to disable the FlexRay module.

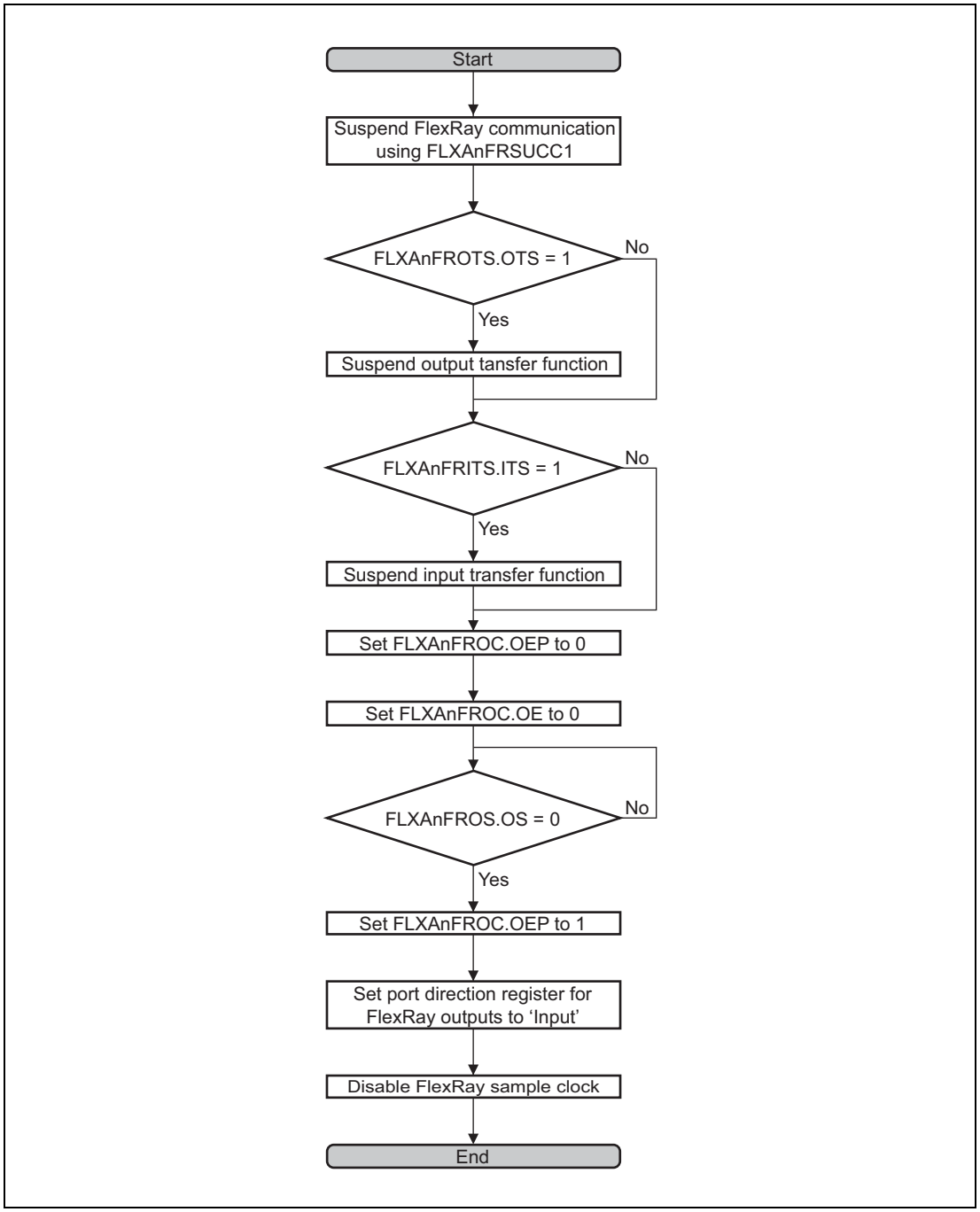



**Figure 20.4 FlexRay disable flow**

### 20.3.2 Communication Cycle

Communication on FlexRay networks is based on frames and symbols. The wakeup symbol (WUS) and the collision avoidance symbol (CAS) are transmitted outside the communication cycle to setup the time schedule. Frames and media access test symbols (MTS) are transmitted inside the communication cycle.

A FlexRay communication cycle consists of the following elements:

- Static Segment
- Dynamic Segment (optional)
- Symbol Window (optional)
- Network Idle Time (NIT)

Static segment, dynamic segment, and symbol window form the Network Communication Time (NCT). For each communication channel the slot counter starts at 1 and counts up until the end of the dynamic segment is reached. Both channels share the same arbitration grid which means that they use the same synchronized macrotick.

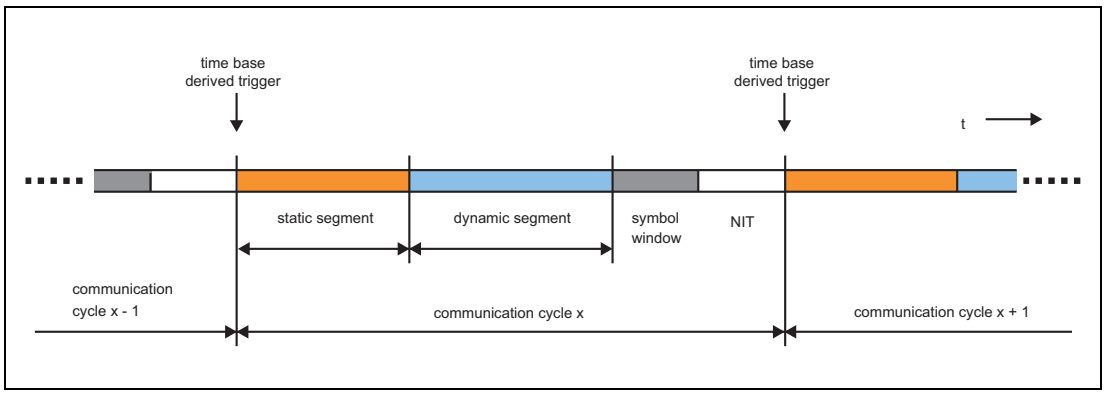



**Figure 20.5 Structure of Communication Cycle**

#### 20.3.2.1 Static Segment

The Static Segment is characterized by the following features:

- Time slots of fixed length (optionally protected by bus guardian)
- Start of frame transmission at action point of the respective static slot
- Payload length same for all frames on both channels

**Parameters:**

Number of Static Slots (FLXAnFRGTUC7.NSS)

Static Slot Length (FLXAnFRGTUC7.SSL)

Payload Length Static (FLXAnFRMHDC.SFDL)

Action Point Offset (FLXAnFRGTUC9.APO)

#### 20.3.2.2 Dynamic Segment

The Dynamic Segment is characterized by the following features:

- All controllers have bus access (no bus guardian protection possible)
- Variable payload length and duration of slots, different for both channels
- Start of transmission at minislot action point

**Parameters:**

Number of Minislots (FLXAnFRGTUC8.NMS)

Minislot Length (FLXAnFRGTUC8.MSL)

Minislot Action Point Offset (FLXAnFRGTUC9.MAPO)

Start of Latest Transmit (last minislot) (FLXAnFRMHDC.SLT)

#### 20.3.2.3 Symbol Window

During the symbol window only one media access test symbol (MTS) may be transmitted per channel. MTS symbols are send in NORMAL_ACTIVE state to test the bus guardian.

The symbol window is characterized by the following features:

- Send single symbol
- Transmission of the MTS symbol starts at the symbol windows action point

**Parameters:**

Symbol Window Action Point Offset (FLXAnFRGTUC9.APO) (same as for static slots)

Network Idle Time Start (FLXAnFRGTUC4.NIT)

#### 20.3.2.4 Network Idle Time (NIT)

During network idle time the CC has to perform the following tasks:

- Calculate clock correction terms (offset and rate)
- Distribute offset correction over multiple macroticks after offset correction start
- Perform cluster cycle related tasks

**Parameters:**

Network idle time start bit (FLXAnFRGTUC4.NIT)

Offset correction start bit (FLXAnFRGTUC4.OCS)

### 20.3.2.5 Configuration of NIT Start and Offset Correction Start

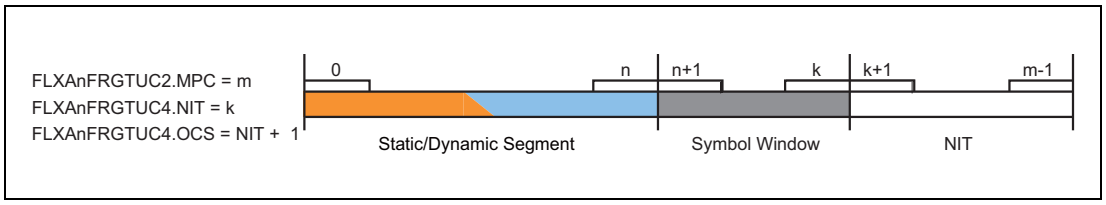



**Figure 20.6 Configuration of NIT Start and Offset Correction Start**

The number of macroticks per cycle gMacroPerCycle is assumed to be m. It is configured by programming FLXAnFRGTUC2.MPC = m.

The static / dynamic segment starts with macrotick 0 and ends with macrotick n:

n = static segment length + dynamic segment offset + dynamic segment length - 1MT

n = gNumberOfStaticSlots * gdStaticSlot + dynamic segment offset + gNumberOfMinislots * gdMinislot - 1 MT

The static segment length is configured by FLXAnFRGTUC7.SSL and FLXAnFRGTUC7.NSS.

The dynamic segment length is configured by FLXAnFRGTUC8.MSL and FLXAnFRGTUC8.NMS.

The dynamic segment offset is:

If gdActionPointOffset <= gdMinislotActionPointOffset:

dynamic segment offset = 0 MT

Else if gdActionPointOffset > gdMinislotActionPointOffset:

dynamic segment offset = gdActionPointOffset -gdMinislotActionPointOffset

The NIT starts with macrotick k + 1 and ends with the last macrotick of cycle m-1. It has to be configured by setting FLXAnFRGTUC4.NIT = k.

For the FlexRay module the offset correction start is required to be FLXAnFRGTUC4.OCS ≥ FLXAnFRGTUC4.NIT + 1 = k + 1.

The length of symbol window results from the number of macroticks between the end of the static / dynamic segment and the beginning of the NIT. It can be calculated by the number of macroticks (k – n).

### 20.3.3 Communication Modes

The FlexRay Protocol Specification defines the Time-Triggered Distributed (TT-D) mode.

#### 20.3.3.1 Time-triggered Distributed (TT-D)

In TT-D mode the following configurations are possible:

- Pure static: Minimum 2 static slots + symbol window (optional)
- Mixed static/dynamic: Minimum 2 static slots + dynamic segment + symbol window (optional)

A minimum of two coldstart nodes needs to be configured for distributed time-triggered operation. Two fault-free coldstart nodes are necessary for the cluster startup. Each startup frame must be a sync frame, therefore all coldstart nodes are sync nodes.

### 20.3.4 Clock Synchronization

In TT-D mode a distributed clock synchronization is used. Each node individually synchronizes itself to the cluster by observing the timing of received sync frames from other nodes.

#### 20.3.4.1 Global Time

Activities in a FlexRay node, including communication, are based on the concept of a global time. It is the clock synchronization mechanism that differentiates the FlexRay cluster from other node collections with independent clock mechanisms. The global time is a vector of two values; the cycle (cycle counter) and the cycle time (macrotick counter).

Cluster specific:

- Macrotick (MT) = basic unit of time measurement in a FlexRay network, a macrotick consists of an integer number of microticks (µT)
- Cycle length = duration of a communication cycle in units of macroticks (MT)

#### 20.3.4.2 Local Time

Internally, nodes time their behavior with microtick resolution. Microticks are time units derived from the oscillator clock tick of the specific node. Therefore microticks are controller-specific units. They may have different duration in different controllers. The precision of a node's local time difference measurements is a microtick (µT).

Node specific:

- Oscillator clock → prescaler → microtick (µT)
- µT = basic unit of time measurement in a CC, clock correction is done in units of µTs
- Cycle counter + macrotick counter = nodes local view of the global time

#### 20.3.4.3 Synchronization Process

Clock synchronization is performed by means of sync frames. Only preconfigured nodes (sync nodes) are allowed to send sync frames. In a two-channel cluster a sync node has to send its sync frame on both channels.

For synchronization in FlexRay the following constraints have to be considered:

- Max. one sync frame per node in one communication cycle
- Max. 15 sync frames per cluster in one communication cycle
- Every node has to use a preconfigured number of sync frames (FLXAnFRGTUC2.SNM) for clock synchronization
- Minimum of two sync nodes required for clock synchronization and startup

For clock synchronization the time difference between expected and observed arrival time of sync frames received during the static segment is measured. In a two channel cluster the sync node has to be configured to send sync frames on both channels. The calculation of correction terms is done during NIT (offset: every cycle, rate: every odd cycle) by using an FTM algorithm. For details see FlexRay protocol specification v2.1, chapter 8.

### (1) Offset (phase) Correction

- Only deviation values measured and stored in the current cycle used
- For a two channel node the smaller value will be taken
- Calculation during NIT of every communication cycle
- Offset correction value calculated in even cycles used for error checking only
- Checked against limit values
- Correction value is a signed integer number of µTs
- Correction done in odd numbered cycles, distributed over the macroticks beginning at offset correction start up to cycle end (end of NIT) to shift nodes next start of cycle (MTs lengthened / shortened)

### (2) Rate (frequency) Correction

- Pairs of deviation values measured and stored in even / odd cycle pair used
- For a two channel node the average of the differences from the two channels is used
- Calculated during NIT of odd numbered cycles
- Cluster drift damping is performed using global damping value
- Checked against limit values
- Correction value is a signed integer number of µTs
- Distributed over macroticks comprising the next even / odd cycle pair (MTs lengthened / shortened)

### (3) Sync Frame Transmission

Sync frame transmission is only possible from buffer 0 and 1. Message buffer 1 may be used for sync frame transmission in case that sync frames should have different payloads on the two channels. In this case bit FLXAnFRMRC.SPLM has to be programmed to 1.

Message buffers used for sync frame transmission have to be configured with the key slot ID and can be (re) configured in DEFAULT_CONFIG or CONFIG state only. For nodes transmitting sync frames FLXAnFRSUCC1.TXSY must be set to 1.

### (4) External Clock Synchronization

During normal operation, independent clusters can drift significantly. If synchronous operation across independent clusters is desired, external synchronization is necessary; even though the nodes within each cluster are synchronized. This can be accomplished with synchronous application of host-deduced rate and offset correction terms to the clusters.

- External offset / rate correction value is a signed integer
- External offset / rate correction value is added to calculated offset / rate correction value
- Aggregated offset / rate correction term (external + internal) is not checked against configured limits

### 20.3.5 Error Handling

The implemented error handling concept is intended to ensure that, in case of a lower layer protocol error in one single node, communication between non-affected nodes can be maintained. In some cases, higher layer program activity is required for the CC to resume normal operation. A change of the error handling state will set FLXAnFREIR.PEMC to 1 and may trigger an interrupt to the Host if enabled. The actual error mode is signaled by FLXAnFRCCEV.ERRM.

**Table 20.109 Error modes of the POC (degradation model)**

| Error Mode | Activity |
|---|---|
| ACTIVE | Full operation, State: NORMAL_ACTIVE<br>The CC is fully synchronized and supports the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status interrupt flags from registers FLXAnFREIR and FLXAnFRSIR. |
| PASSIVE | Reduced operation, State: NORMAL_PASSIVE, CC self rescue allowed<br>The CC stops transmitting frames and symbols, but received frames are still processed. Clock synchronization mechanisms are continued based on received frames. No active contribution to the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status interrupt flags from registers FLXAnFREIR and FLXAnFRSIR. |
| COMM_HALT | Operation halted, State: HALT, CC self rescue not allowed<br>The CC stops frame and symbol processing, clock synchronization processing, and the macrotick generation. The host has still access to error and status information by reading the error and status interrupt flags from registers FLXAnFREIR and FLXAnFRSIR. The bus drivers are disabled. |

#### 20.3.5.1 Clock Correction Failed Counter

When the Clock Correction Failed Counter reaches the “maximum without clock correction passive” limit defined by FLXAnFRSUCC3.WCP, the POC transits from NORMAL_ACTIVE to NORMAL_PASSIVE state. When it reaches the “maximum without clock correction fatal” limit defined by FLXAnFRSUCC3.WCF, it transits from NORMAL_ACTIVE or NORMAL_PASSIVE to HALT state.

The Clock Correction Failed Counter FLXAnFRCCEV.CCFC allows the Host to monitor the duration of the inability of a node to compute clock correction terms after the CC passed protocol startup phase. It will be incremented by one at the end of any odd communication cycle during which either the missing offset correction FLXAnFRSFS.MOCS or the missing rate correction FLXAnFRSFS.MRCS flag is set to 1.

The Clock Correction Failed Counter is reset to zero at the end of an odd communication cycle if neither the missing offset correction FLXAnFRSFS.MOCS nor the missing rate correction FLXAnFRSFS.MRCS flag is set to 1.

The Clock Correction Failed Counter stops incrementing when the “maximum without clock correction fatal” value FLXAnFRSUCC3.WCF is reached (i.e. incrementing the counter at its maximum value will not cause it to wrap around back to zero). The Clock Correction Failed Counter is initialized to zero when the CC enters READY state or when NORMAL_ACTIVE state is entered.

**CAUTION**

**The transition to HALT state is prevented if FLXAnFRSUCC1.HCSE is not set to 1.**

#### 20.3.5.2 Passive to Active Counter

The passive to active counter controls the transition of the POC from NORMAL_PASSIVE to NORMAL_ACTIVE state. FLXAnFRSUCC1.PTA defines the number of consecutive even / odd cycle pairs that must have valid clock correction terms before the CC is allowed to transit from NORMAL_PASSIVE to NORMAL_ACTIVE state. If FLXAnFRSUCC1.PTA is set to zero the CC is not allowed to transit from NORMAL_PASSIVE to NORMAL_ACTIVE state.

#### 20.3.5.3 HALT Command

In case the Host wants to stop FlexRay communication of the local node it can bring the CC into HALT state by asserting the HALT command. This can be done by writing FLXAnFRSUCC1.CMD = “0110”. In order to shut down communication on an entire FlexRay network, a higher layer protocol is required to assure that all nodes apply the HALT command at the same time.

The POC state from which the transition to HALT state took place can be read from FLXAnFRCCSV.PSL.

When called in NORMAL_ACTIVE or NORMAL_PASSIVE state the POC transits to HALT state at the end of the current cycle. When called in any other state FLXAnFRSUCC1.CMD will be reset to “0000” = command_not_accepted and bit FLXAnFREIR.CNA is set to 1. If enabled an interrupt to the Host is generated.

#### 20.3.5.4 FREEZE Command

In case the Host detects a severe error condition it can bring the CC into HALT state by asserting the FREEZE command. This can be done by writing FLXAnFRSUCC1.CMD = “0111”. The FREEZE command triggers the entry of the HALT state immediately regardless of the actual POC state.

The POC state from which the transition to HALT state took place can be read from FLXAnFRCCSV.PSL.

**CAUTION**

**When the user restart as leading coldstarter after it has been stopped by FREEZE or READY command, it may happen, depending on the internal state of the FlexRay module, that the FlexRay module does not transmit its startup frame in cycle 0. Only FlexRay module configurations with startup frames configured for slots 1 to 7 are affected by this behaviour.**

**Coldstart after hardware reset is not affected.**

**Even if it happened, next coldstart attempt successful. Coldstart sequence is lengthened but coldstart of FlexRay system is not prohibited by this behaviour.**

**If the use wants to avoid this behavior, they must use a static slot greater or equal 8 for the startup / sync message.**

## 20.3.6 Communication Controller States

### 20.3.6.1 Communication Controller State Diagram

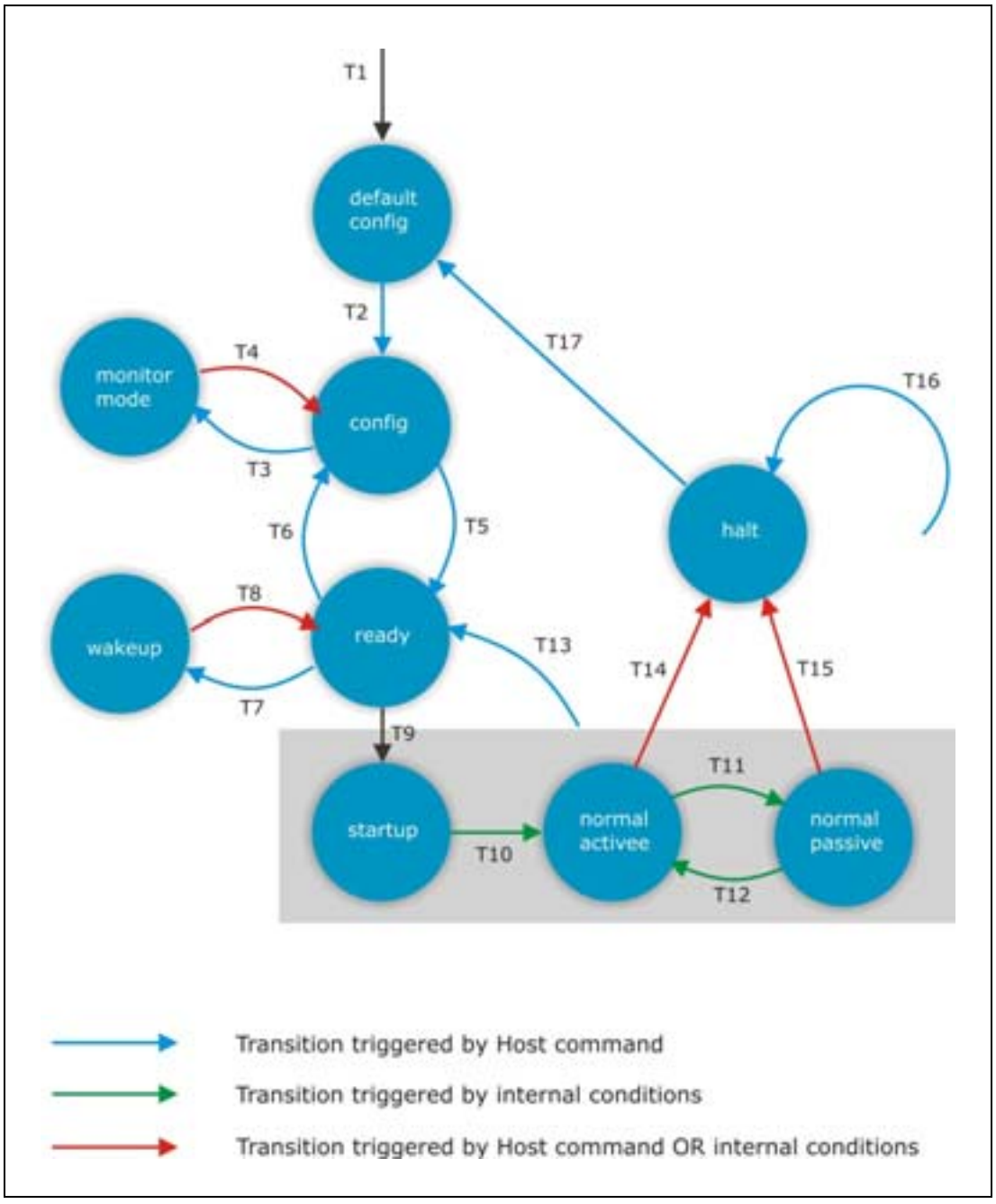



**Figure 20.7 Overall State Diagram of FlexRay Communication Controller**

State transitions are controlled by reset, FLXAnFR0RXDA, FLXAnFR0RXDB, by the POC state machine, and by the CHI Command Vector FLXAnFRSUCC1.CMD.

The CC transits from all states to HALT state after application of the FREEZE command (FLXAnFRSUCC1.CMD = “0111”).

**Table 20.110 State Transitions of FlexRay Overall State Machine**

| T# | Condition | From | To |
|---|---|---|---|
| 1 | Reset | All States | DEFAULT_CONFIG |
| 2 | Command CONFIG, FLXAnFRSUCC1.CMD = “0001” | DEFAULT_CONFIG | CONFIG |
| 3 | Unlock sequence followed by command MONITOR_MODE, FLXAnFRSUCC1.CMD[3:0] = “1011” | CONFIG | MONITOR_MODE |
| 4 | Command CONFIG, FLXAnFRSUCC1.CMD = "0001" | MONITOR_MODE | CONFIG |
| 5 | Unlock sequence followed by command READY, FLXAnFRSUCC1.CMD = “0010” | CONFIG | READY |
| 6 | Command CONFIG, FLXAnFRSUCC1.CMD = “0001” | READY | CONFIG |
| 7 | Command WAKEUP, FLXAnFRSUCC1.CMD = “0011” | READY | WAKEUP |
| 8 | Complete, non-aborted transmission of wakeup pattern OR received WUP OR received frame header OR wakeup collision OR command READY, FLXAnFRSUCC1.CMD = “0010” | WAKEUP | READY |
| 9 | Command RUN, FLXAnFRSUCC1.CMD = “0100” | READY | STARTUP |
| 10 | Successful startup | STARTUP | NORMAL_ACTIVE |
| 11 | Clock Correction Failed counter reached Maximum Without Clock Correction Passive limit configured by FLXAnFRSUCC3.WCP | NORMAL_ACTIVE | NORMAL_PASSIVE |
| 12 | Number of valid correction terms reached the Passive to Active limit configured by FLXAnFRSUCC1.PTA | NORMAL_PASSIVE | NORMAL_ACTIVE |
| 13 | Command READY, FLXAnFRSUCC1.CMD = “0010” | STARTUP, NORMAL_ACTIVE, NORMAL_PASSIVE | READY |
| 14 | Clock Correction Failed counter reached Maximum Without Clock Correction Fatal limit configured by FLXAnFRSUCC3.WCF AND bit FLXAnFRSUCC1.HCSE set to ’1’ OR command HALT, FLXAnFRSUCC1.CMD = “0110” | NORMAL_ACTIVE | HALT |
| 15 | Clock Correction Failed counter reached Maximum Without Clock Correction Fatal limit configured by FLXAnFRSUCC3.WCF AND bit FLXAnFRSUCC1.HCSE set to ’1’ OR command HALT, FLXAnFRSUCC1.CMD = “0110” | NORMAL_PASSIVE | HALT |
| 16 | Command FREEZE, FLXAnFRSUCC1.CMD = “0111” | All States | HALT |
| 17 | Command CONFIG, FLXAnFRSUCC1.CMD = “0001” | HALT | DEFAULT_CONFIG |

#### 20.3.6.2 DEFAULT_CONFIG State

In DEFAULT_CONFIG state, the CC is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state.

The CC enters this state

- When the reset is applied (HW reset or SW reset)
- When exiting from HALT state

To leave DEFAULT_CONFIG state the Host has to write FLXAnFRSUCC1.CMD = “0001”. The CC then transits to CONFIG state.

#### 20.3.6.3 CONFIG State

In CONFIG state, the CC is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state. This state is used to initialize the CC configuration.

The CC enters this state

- When exiting from DEFAULT_CONFIG state
- When exiting from READY state

When the state has been entered via HALT and DEFAULT_CONFIG state, the Host can analyse status information and configuration. Before leaving CONFIG state the Host has to assure that the configuration is fault-free.

To leave CONFIG state, the Host has to perform the unlock sequence as described is **Section 20.2.3.1, FLXAnFRLCK — FlexRay Lock Register**. Directly after unlocking the CONFIG state the Host has to write FLXAnFRSUCC1.CMD to enter the next state.

**CAUTION**

**Status bits FLXAnFRMHDS[14:0], registers FLXAnFRTXRQ1/2/3/4, and status data stored in the Message RAM are not affected by the transition of the POC from CONFIG to READY state.**

When the CC is in CONFIG state it is also possible to bring the CC into a power saving mode by halting the module clocks (bus clock and sample clock). To do this the Host has to assure that all Message RAM transfers have finished before turning off the clocks.

#### 20.3.6.4 READY State

After unlocking CONFIG state and writing FLXAnFRSUCC1.CMD = “0010” the CC enters READY state. From this state the CC can transit to WAKEUP state and perform a cluster wakeup or to STARTUP state to perform a coldstart or to integrate into a running cluster.

The CC enters this state

- When exiting from CONFIG, WAKEUP, STARTUP, NORMAL_ACTIVE, or NORMAL_PASSIVE state by writing FLXAnFRSUCC1.CMD = “$0010_B$” (READY command).

The CC exits from this state

- To CONFIG state by writing FLXAnFRSUCC1.CMD = “$0001_B$” (CONFIG command)
- To WAKEUP state by writing FLXAnFRSUCC1.CMD = “$0011_B$” (WAKEUP command)
- To STARTUP state by writing FLXAnFRSUCC1.CMD = “$0100_B$” (RUN command)

Internal counters and the CC status flags are reset when the CC enters STARTUP state.

**CAUTION**

**Status bits FLXAnFRMHDS[14:0], registers FLXAnFRTXRQ1/2/3/4, and status data stored in the Message RAM are not affected by the transition of the POC from READY to STARTUP state.**

### 20.3.6.5 WAKEUP State

The description below is intended to help configuring wakeup for the FlexRay IP-module. A detailed description of the wakeup procedure together with the respective SDL diagrams can be found in the FlexRay protocol specification v2.1, **Section 7.1**.

The CC enters this state

- When exiting from READY state by writing FLXAnFRSUCC1.CMD = "$0011_B$" (WAKEUP command).

The CC exits from this state to READY state

- After complete non-aborted transmission of wakeup pattern
- After WUP reception
- After detecting a WUP collision
- After reception of a frame header
- By writing FLXAnFRSUCC1.CMD = "$0010_B$" (READY command)

The cluster wakeup must precede the communication startup in order to ensure that all nodes in a cluster are awake. The minimum requirement for a cluster wakeup is that all bus drivers are supplied with power. A bus driver has the ability to wake up the other components of its node when it receives a wakeup pattern on its channel. At least one node in the cluster needs an external wakeup source.

The Host completely controls the wakeup procedure. It is informed about the state of the cluster by the bus driver and the CC and configures bus guardian (if available) and CC to perform the cluster wakeup. The CC provides to the Host the ability to transmit a special wakeup pattern on each of its available channels separately. The CC needs to recognize the wakeup pattern only during WAKEUP state.

Wakeup may be performed on only one channel at a time. The Host has to configure the wakeup channel while the CC is in CONFIG state by writing FLXAnFRSUCC1.WUCS. The CC ensures that ongoing communication on this channel is not disturbed. The CC cannot guarantee that all nodes connected to the configured channel awake upon the transmission of the wakeup pattern, since these nodes cannot give feedback until the startup phase. The wakeup procedure enables single-channel devices in a two-channel system to trigger the wakeup, by only transmitting the wakeup pattern on the single channel to which they are connected. Any coldstart node that deems a system startup necessary will then wake the remaining channel before initiating communication startup.

The wakeup procedure tolerates any number of nodes simultaneously trying to wakeup a single channel and resolves this situation such that only one node transmits the pattern. Additionally the wakeup pattern is collision resilient, so even in the presence of a fault causing two nodes to simultaneously transmit a wakeup pattern, the resulting collided signal can still wake the other nodes.

After wakeup the CC returns to READY state and signals the change of the wakeup status to the Host by setting flag FLXAnFRSIR.WST. The wakeup status vector can be read from FLXAnFRCCSV.WSV. If a valid wakeup pattern was received also either flag FLXAnFRSIR.WUPA or flag FLXAnFRSIR.WUPB is set to 1.

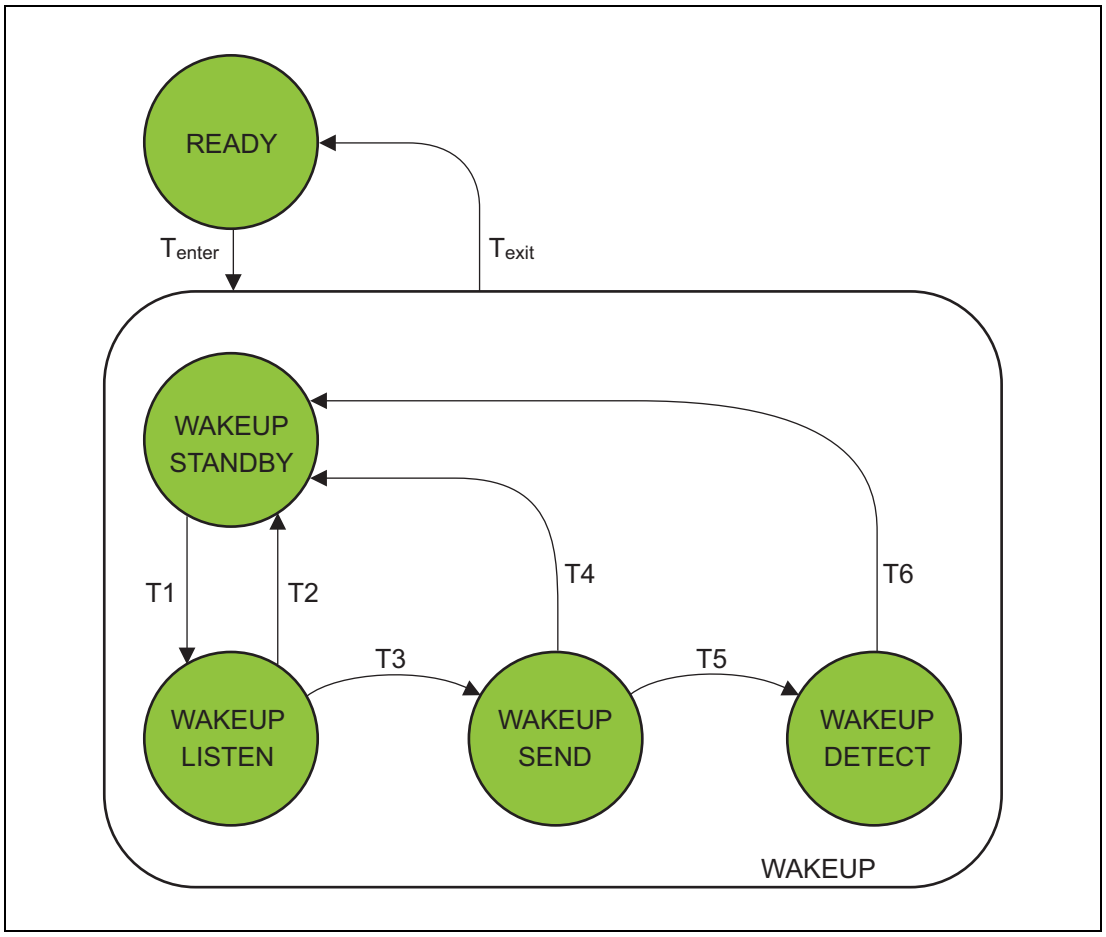



**Figure 20.8 Structure of POC State WAKEUP**

**Table 20.111 State Transitions WAKEUP (1/2)**

| T# | Condition | From | To |
|---|---|---|---|
| enter | Host commands change to WAKEUP state by writing FLXAnFRSUCC1.CMD = “0011” (WAKEUP command) | READY | WAKEUP |
| 1 | CHI command WAKEUP triggers wakeup FSM to transit to WAKEUP_LISTEN state | WAKEUP_STANDBY | WAKEUP_LISTEN |
| 2 | Received WUP on wakeup channel selected by bit FLXAnFRSUCC1.WUCS or frame header on either available channel | WAKEUP_LISTEN | WAKEUP_STANDBY |
| 3 | Timer event | WAKEUP_LISTEN | WAKEUP_SEND |
| 4 | Complete, non-aborted transmission of wakeup pattern | WAKEUP_SEND | WAKEUP_STANDBY |
| 5 | Collision detected | WAKEUP_SEND | WAKEUP_DETECT |
| 6 | Wakeup timer expired or WUP detected on wakeup channel selected by bit FLXAnFRSUCC1.WUCS or frame header received on either available | WAKEUP_DETECT | WAKEUP_STANDBY |

Table 20.111 State Transitions WAKEUP (2/2)

| T# | Condition | From | To |
|---|---|---|---|
| exit | Wakeup completed (after T2 or T4 or T6) or Host commands change to READY state by writing FLXAnFRSUCC1.CMD = “0010” (READY command). This command also resets the wakeup FSM to WAKEUP_STANDBY state | WAKEUP | READY |

The WAKEUP_LISTEN state is controlled by the wakeup timer and the wakeup noise timer. The two timers are controlled by the parameters listen timeout FLXAnFRSUCC2.LT and listen timeout noise FLXAnFRSUCC2.LTN. Listen timeout enables a fast cluster wakeup in case of a noise free environment, while listen timeout noise enables wakeup under more difficult conditions regarding noise interference.

In WAKEUP_SEND state the CC transmits the wakeup pattern on the configured channel and checks for collisions. After return from wakeup the Host has to bring the CC into STARTUP state by CHI command RUN.

In WAKEUP_DETECT state the CC attempts to identify the reason for the wakeup collision detected in WAKEUP_SEND state. The monitoring is bounded by the expiration of listen timeout as configured by FLXAnFRSUCC2.LT. Either the detection of a wakeup pattern indicating a wakeup attempt by another node or the reception of a frame header indicating ongoing communication, causes the direct transition to READY state. Otherwise WAKEUP_DETECT is left after expiration of listen timeout; in this case the reason for wakeup collision is unknown.

The Host has to be aware of possible failures of the wakeup and act accordingly. It is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal time it takes another coldstart node to become awake and to be configured.

The FlexRay Protocol Specification recommends that two different CCs shall awake the two channels.

### (1) Host activities

The host must coordinate the wakeup of the two channels and must decide whether, or not, to wake a specific channel. The sending of the wakeup pattern is initiated by the Host. The wakeup pattern is detected by the remote BDs and signaled to their local Host.

**Wakeup procedure controlled by Host (single-channel wakeup):**

- Configure the CC in CONFIG state
  - Select wakeup channel by programming bit FLXAnFRSUCC1.WUCS
- Check local BDs whether a WUP was received
- Activate BD of selected wakeup channel
- Command CC to enter READY state
- Command CC to start wakeup on the configured channel by writing FLXAnFRSUCC1.CMD = “0011”
  - CC enters WAKEUP
  - CC returns to READY state and signals status of wakeup attempt to the Host
- Wait predefined time to allow the other nodes to wakeup and configure themselves
- Coldstart node:

  - In a dual channel cluster wait for WUP on the other channel
  - Reset coldstart inhibit flag FLXAnFRCCSV.CSI by writing FLXAnFRSUCC1.CMD = “1001” (ALLOW_COLDSTART command)
- Command CC to enter startup by writing FLXAnFRSUCC1.CMD = “0100” (RUN command)

**Wakeup procedure triggered by BD:**

- Wakeup recognized by BD
- BD triggers power-up of Host (if required)
- BD signals wakeup event to Host
- Host configures its local CC
- If necessary, Host commands wakeup of second channel and waits predefined time to allow the other nodes to wakeup and configure themselves
- Host commands CC to enter STARTUP state by writing FLXAnFRSUCC1.CMD = “0100” (RUN command)

### (2) Wakeup pattern (WUP)

The wakeup pattern (WUP) is composed of at least two wakeup symbols (WUS). Wakeup symbol and wakeup pattern are configured by registers FLXAnFRPRTC1 and FLXAnFRPRTC2.

- Single channel wakeup, wakeup symbol may not be sent on both channels at the same time
- Wakeup symbol collision resilient for at least two sending nodes (two overlapping wakeup symbols always recognizable)
- Wakeup symbol must be configured identical in all nodes of a cluster
- Wakeup symbol transmit low time configured by FLXAnFRPRTC2.TXL
- Wakeup symbol idle time used to listen for activity on the bus, configured by FLXAnFRPRTC2.TXI
- A wakeup pattern composed of at least two Tx-wakeup symbols needed for wakeup
- Number of repetitions configurable by FLXAnFRPRTC1.RWP (2 to 63 repetitions)
- Wakeup symbol receive window length configured by FLXAnFRPRTC1.RXW
- Wakeup symbol receive low time configured by FLXAnFRPRTC2.RXL
- Wakeup symbol receive idle time configured by FLXAnFRPRTC2.RXI

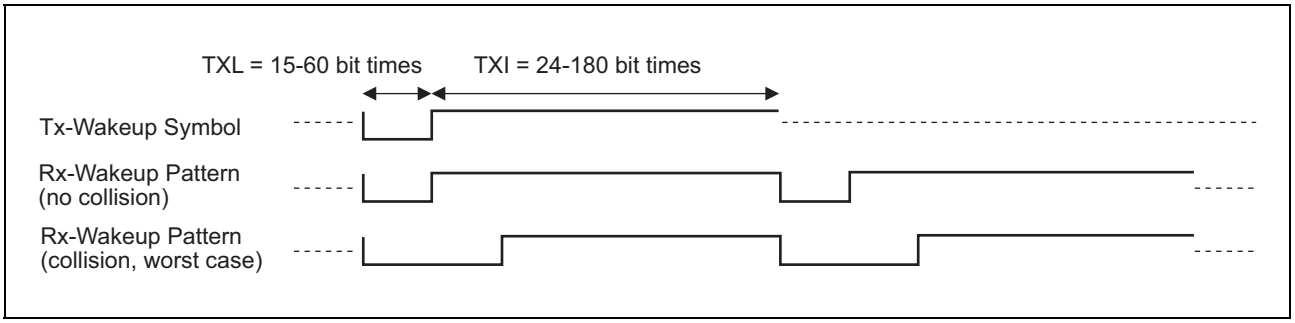



**Figure 20.9 Timing of Wakeup Pattern**

#### 20.3.6.6 STARTUP State

The description below is intended to help configuring startup for the FlexRay IP-module. A detailed description of the startup procedure together with the respective SDL diagrams can be found in the FlexRay protocol specification v2.1, **Section 7.2**.

Any node entering STARTUP state that has coldstart capability should assure that both channels attached have been awakened before initiating coldstart.

It cannot be assumed that all nodes and stars need the same amount of time to become completely awake and to be configured. Since at least two nodes are necessary to start up the cluster communication, it is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal amount of time it takes another coldstart node to become awake, to be configured and to enter startup. It may require several hundred milliseconds (depending on the hardware used) before all nodes and stars are completely awakened and configured.

Startup is performed on all channels synchronously. During startup, a node only transmits startup frames. Startup frames are both sync frames and null frames during startup.

A fault-tolerant, distributed startup strategy is specified for initial synchronization of all nodes. In general, a node may enter NORMAL_ACTIVE state via (see **Figure 20.10**):

- Coldstart path initiating the schedule synchronization (LeadingColdstart node)
- Coldstart path joining other coldstart nodes (FollowingColdstart node)
- Integration path integrating into an existing communication schedule (all other nodes)

A coldstart attempt begins with the transmission of a collision avoidance symbol (CAS). Only a coldstart node that had transmitted the CAS transmits frames in the first four cycles after the CAS, it is then joined firstly by the other coldstart nodes and afterwards by all other nodes.

A coldstart node has bits FLXAnFRSUCC1.TXST and FLXAnFRSUCC1.TXSY set to 1. Message buffer 0 holds the key slot ID which defines the slot number where the startup frame is send. In the frame header of the startup frame the startup frame indicator bit is set to 1.

In clusters consisting of three or more nodes, at least three nodes shall be configured to be coldstart nodes. In clusters consisting of two nodes, both nodes must be coldstart nodes. At least two fault-free coldstart nodes are necessary for the cluster to startup.

Each startup frame must also be a sync frame; therefore each coldstart node will also be a sync node. The number of coldstart attempts is configured by FLXAnFRSUCC1.CSA.

A non-coldstart node requires at least two startup frames from distinct nodes for integration. It may start integration before the coldstart nodes have finished their startup. It will not finish its startup until at least two coldstart nodes have finished their startup.

Both non-coldstart nodes and coldstart nodes start passive integration via the integration path as soon as they receive sync frames from which to derive the TDMA schedule information. During integration, the node has to adapt its own clock to the global clock (rate and offset) and has to make its cycle time consistent with the global schedule observable at the network. Afterwards, these settings are checked for consistency with all available network nodes. The node can only leave the integration phase and actively participate in communication when these checks are passed.

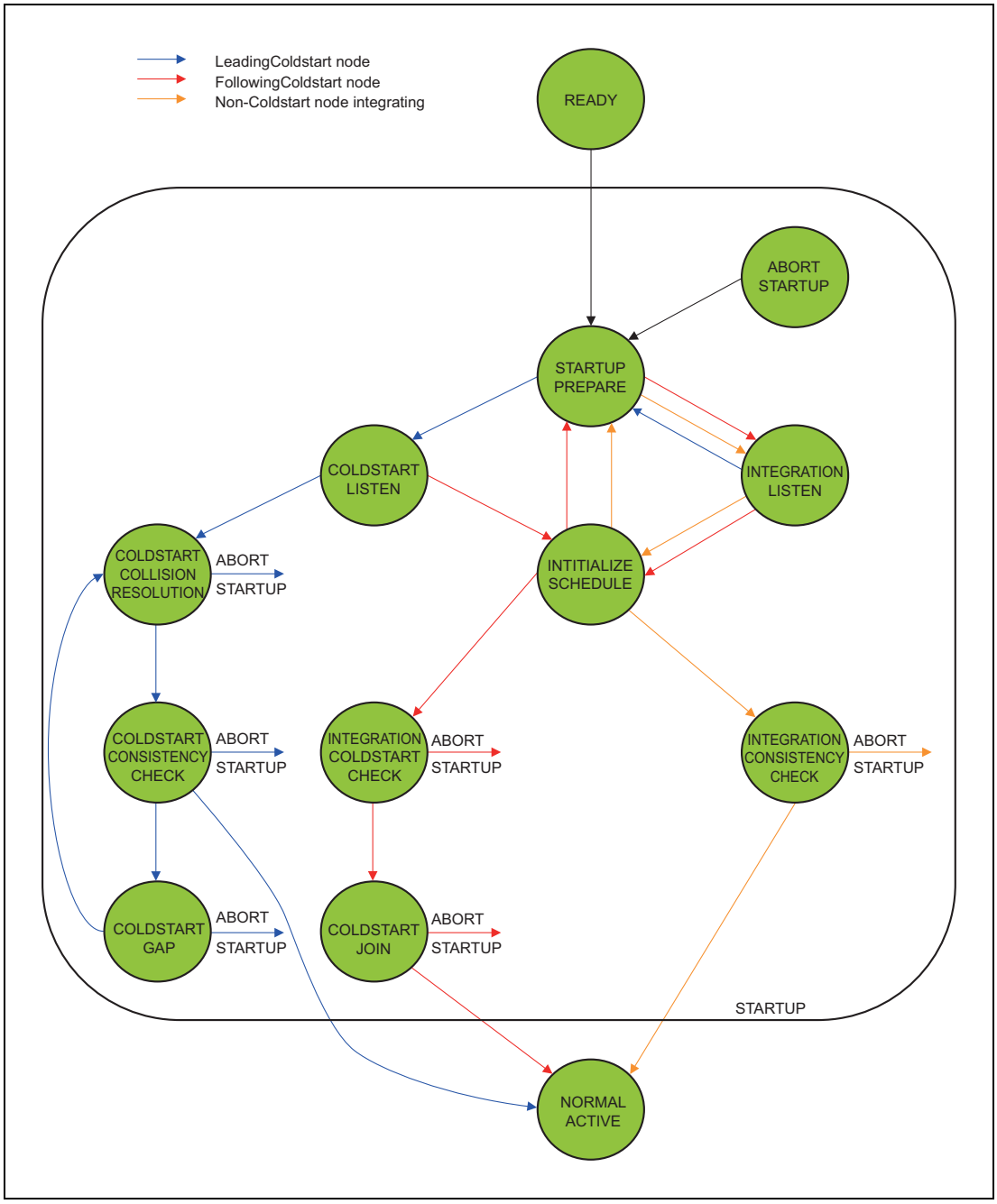



**Figure 20.10 State Diagram Time-Triggered Startup**

### (1) Coldstart Inhibit Mode

In coldstart inhibit mode the node is prevented from initializing the TDMA communication schedule. If bit FLXAnFRCCSV.CSI is set to 1, the node is not allowed to initialize the cluster communication, i.e. entering the coldstart path is prohibited. The node is allowed to integrate to a running cluster or to transmit startup frames after another coldstart node started the initialization of the cluster communication.

The coldstart inhibit bit FLXAnFRCCSV.CSI is set to 1 whenever the POC enters READY state. The bit has to be cleared under control of the Host by CHI command ALLOW_COLDSTART (FLXAnFRSUCC1.CMD = “1001”)

### (2) Startup Timeouts

The CC supplies two different µT timers supporting two timeout values, startup timeout and startup noise timeout. The two timers are started when the CC enters the COLDSTART_LISTEN state. The expiration of either of these timers causes the node to leave the initial sensing phase (COLDSTART_LISTEN state) with the intention of starting up communication.

**CAUTION**

**The startup and startup noise timers are identical with the wakeup and wakeup noise timers and use the same configuration values FLXAnFRSUCC2.LT and FLXAnFRSUCC2.LTN.**

#### (a) Startup Timeout

The startup timeout limits the listen time used by a node to determine if there is already communication between other nodes or at least one coldstart node actively requesting the integration of others. The startup timer is configured by programming FLXAnFRSUCC2.LT (see **Section 20.2.6.2, FLXAnFRSUCC2 — FlexRay SUC Configuration Register 2**).

The startup timeout is:

pdListenTimeout = FLXAnFRSUCC2.LT

The startup timer is restarted upon:

- Entering the COLDSTART_LISTEN state
- Both channels reaching idle state while in COLDSTART_LISTEN state

The startup timer is stopped:

- If communication channel activity is detected on one of the configured channels while the node is in the COLDSTART_LISTEN state
- When the COLDSTART_LISTEN state is left

Once the startup timeout expires, neither an overflow nor a cyclic restart of the timer is performed. The timer status is kept for further processing by the startup state machine.

#### (b) Startup Noise Timeout

At the same time the startup timer is started for the first time (transition from STARTUP_PREPARE state to COLDSTART_LISTEN state), the startup noise timer is started. This additional timeout is used to improve reliability of the startup procedure in the presence of noise. The startup noise timeout is configured by programming FLXAnFRSUCC2.LTN (see **Section 20.2.6.2, FLXAnFRSUCC2 — FlexRay SUC Configuration Register 2**).

The startup noise timeout is:

pdListenTimeout * gListenNoise = FLXAnFRSUCC2.LT * (FLXAnFRSUCC2.LTN + 1)

The startup noise timer is restarted upon:

- Entering the COLDSTART_LISTEN state
- Reception of correctly decoded headers or CAS symbols while the node is in COLDSTART_LISTEN state

The startup noise timer is stopped when the COLDSTART_LISTEN state is left.

Once the startup noise timeout expires, neither an overflow nor a cyclic restart of the timer is performed. The status is kept for further processing by the startup state machine. Since the startup noise timer won't be restarted when random channel activity is sensed, this timeout defines the fall-back solution that guarantees that a node will try to start up the communication cluster even in the presence of noise.

### (3) Path of Leading Coldstart Node (initiating coldstart)

When a coldstart node enters COLDSTART_LISTEN, it listens to its attached channels.

If no communication is detected, the node enters the COLDSTART_COLLISION_RESOLUTION state and commences a coldstart attempt. The initial transmission of a CAS symbol is succeeded by the first regular cycle. This cycle has the number zero.

From cycle zero on, the node transmits its startup frame. Since each coldstart node may perform a coldstart attempt, it may occur that several nodes simultaneously transmit the CAS symbol and enter the coldstart path. This situation is resolved during the first four cycles after CAS transmission.

As soon as a node that initiates a coldstart attempt receives a CAS symbol or a frame header during these four cycles, it re-enters the COLDSTART_LISTEN state. Thereby, only one node remains in this path. In cycle four, other coldstart nodes begin to transmit their startup frames.

After four cycles in COLDSTART_COLLISION_RESOLUTION state, the node that initiated the coldstart enters the COLDSTART_CONSISTENCY_CHECK state. It collects all startup frames from cycle four and five and performs the clock correction. If the clock correction does not deliver any errors and it has received at least one valid startup frame pair, the node leaves COLDSTART_CONSISTENCY_CHECK and enters NORMAL_ACTIVE state.

The number of coldstart attempts that a node is allowed to perform is configured by FLXAnFRSUCC1.CSA. The number of remaining coldstarts attempts can be read from FLXAnFRCCSV.RCA. The number of remaining coldstart attempts is reduced by one for each attempted coldstart. A node may enter the COLDSTART_LISTEN state only if this value is larger than one and it may enter the COLDSTART_COLLISION_RESOLUTION state only if this value is larger than zero. If the number of coldstart attempts is one, coldstart is inhibited but integration is still possible.

### (4) Path of Following Coldstart Node (responding to Leading Coldstart Node)

When a coldstart node enters the COLDSTART_LISTEN state, it tries to receive a valid pair of startup frames to derive its schedule and clock correction from the leading coldstart node.

As soon as a valid startup frame has been received the INITIALIZE_SCHEDULE state is entered. If the clock synchronization can successfully receive a matching second valid startup frame and derive a schedule from this, the INTEGRATION_COLDSTART_CHECK state is entered.

In INTEGRATION_COLDSTART_CHECK state it is assured that the clock correction can be performed correctly and that the coldstart node from which this node has initialized its schedule is still available. The node collects all sync frames and performs clock correction in the following double-cycle. If clock correction does not signal any errors and if the node continues to receive sufficient frames from the same node it has integrated on, the COLDSTART_JOIN state is entered.

In COLDSTART_JOIN state following coldstart nodes begin to transmit their own startup frames and continue to do so in subsequent cycles. Thereby, the leading coldstart node and the nodes joining it can check if their schedules agree with each other. If the clock correction signals any error, the node aborts the integration attempt. If a node in this state sees at least one valid startup frame during all even cycles in this state and at least one valid startup frame pair during all double cycles in this state, the node

leaves COLDSTART_JOIN state and enters NORMAL_ACTIVE state. Thereby it leaves STARTUP at least one cycle after the node that initiated the coldstart.

### (5) Path of Non-coldstart Node

When a non-coldstart node enters the INTEGRATION_LISTEN state, it listens to its attached channels.

As soon as a valid startup frame has been received, the INITIALIZE_SCHEDULE state is entered. If the clock synchronization can successfully receive a matching second valid startup frame and derive a schedule from this, the INTEGRATION_CONSISTENCY_CHECK state is entered.

In INTEGRATION_CONSISTENCY_CHECK state the node verifies that the clock correction can be performed correctly and that enough coldstart nodes (at least 2) are sending startup frames that agree with the node's own schedule. Clock correction is activated, and if any errors are signaled, the integration attempt is aborted.

During the first even cycle in this state, either two valid startup frames or the startup frame of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

During the first double-cycle in this state, either two valid startup frame pairs or the startup frame pair of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

If after the first double-cycle less than two valid startup frames are received within an even cycle, or less than two valid startup frame pairs are received within a double-cycle, the startup attempt is aborted.

Nodes in this state need to see two valid startup frame pairs for two consecutive double-cycles each to be allowed to leave STARTUP and enter NORMAL_OPERATION. Consequently, they leave startup at least one double-cycle after the node that initiated the coldstart and only at the end of a cycle with an odd cycle number.

### 20.3.6.7 NORMAL_ACTIVE State

As soon as the node that transmitted the first CAS symbol (resolving the potential access conflict and entering STARTUP via coldstart path) and one additional node have entered the NORMAL_ACTIVE state, the startup phase for the cluster has finished. In the NORMAL_ACTIVE state, all configured messages are scheduled for transmission. This includes all data frames as well as the sync frames. Rate and offset measurement is started in all even cycles (even / odd cycle pairs required).

In NORMAL_ACTIVE state the CC supports regular communication functions

- The CC performs transmissions and reception on the FlexRay bus as configured
- Clock synchronization is running
- The Host interface is operational

The CC exits from that state to

- HALT state by writing FLXAnFRSUCC1.CMD = "$0110_B$" (HALT command, at the end of the current cycle)
- HALT state by writing FLXAnFRSUCC1.CMD = "$0111_B$" (FREEZE command, immediately)
- HALT state due to change of the error state from ACTIVE to COMM_HALT
- NORMAL_PASSIVE state due to change of the error state from ACTIVE to PASSIVE
- READY state by writing FLXAnFRSUCC1.CMD = "$0010_B$" (READY command)

### 20.3.6.8 NORMAL_PASSIVE State

NORMAL_PASSIVE state is entered from NORMAL_ACTIVE state when the error state changes from ACTIVE to PASSIVE.

In NORMAL_PASSIVE state, the node is able to receive all frames (node is fully synchronized and performs clock synchronization). Contrary to the NORMAL_ACTIVE state, the node does not actively participate in communication, i.e. neither symbols nor frames are transmitted.

In NORMAL_PASSIVE state

- The CC performs reception on the FlexRay bus
- The CC does not transmit any frames or symbols on the FlexRay bus
- Clock synchronization is running
- The Host interface is operational

The CC exits from this state to

- HALT state by writing FLXAnFRSUCC1.CMD = "0110" (HALT command, at the end of the current cycle)
- HALT state by writing FLXAnFRSUCC1.CMD = "0111" (FREEZE command, immediately)
- HALT state due to change of the error state from PASSIVE to COMM_HALT
- NORMAL_ACTIVE state due to change of the error state from PASSIVE to ACTIVE. The transition takes place when FLXAnFRCCEV.PTAC equals FLXAnFRSUCC1.PTA - 1
- To READY state by writing FLXAnFRSUCC1.CMD = "0010" (READY command)

#### 20.3.6.9 HALT State

In this state all communication (reception and transmission) is stopped.

The CC enters this state

- By writing FLXAnFRSUCC1.CMD = "$0110_B$" (HALT command) while the CC is in NORMAL_ACTIVE or NORMAL_PASSIVE state
- By writing FLXAnFRSUCC1.CMD = "$0111_B$" (FREEZE command) from all states
- When exiting from NORMAL_ACTIVE state because the clock correction failed counter reached the "maximum without clock correction fatal" limit and FLXAnFRSUCC1.HCSE is set to 1
- When exiting from NORMAL_PASSIVE state because the clock correction failed counter reached the "maximum without clock correction fatal" limit and FLXAnFRSUCC1.HCSE is set to 1

The CC exits from this state to DEFAULT_CONFIG state

- By writing FLXAnFRSUCC1.CMD = "$0001_B$" (CONFIG command)

When the CC enters HALT state, all configuration and status data is maintained for analysing purposes.

When the Host writes FLXAnFRSUCC1.CMD = "$0110_B$" (HALT command), the CC sets bit FLXAnFRCCSV.HRQ to 1 and enters HALT state at the next end of cycle.

When the Host writes FLXAnFRSUCC1.CMD = "$0111_B$" (FREEZE command), the CC enters HALT state immediately and sets bit FLXAnFRCCSV.FSI to 1.

The POC state from which the transition to HALT state took place can be read from FLXAnFRCCSV.PSL.

### 20.3.7 Network Management

The accrued Network Management (NM) vector can be read from registers FLXAnFRNMV1…3. The CC performs a bit-wise OR operation over all NM vectors out of all received valid NM frames with the Payload Preamble Indicator (PPI) bit set. Only static frames may be configured to hold NM information. The CC updates the NM vector at the end of each cycle.

The length of the NM vector can be configured from 0 to 12 bytes by FLXAnFRNEMC.NML. The NM vector length must be configured identically in all nodes of a cluster.

To configure a transmit buffer to send FlexRay frames with the PPI bit set as 1, bit PPIT in the header section of the respective transmit buffer has to be set to 1 via FLXAnFRWRHS1.PPIT. In addition the Host has to write the NM information to the data section of the respective transmit buffer.

The evaluation of the NM vector has to be done by the application running on the Host.

**Section 20.3.17, Byte Alignment**, for byte alignment of the received NM vector in registers FLXAnFRNMV1 to FLXAnFRNMV3.

**CAUTIONS**

1. **In case a message buffer is configured for transmission / reception of network management frames, the payload length configured in header 2 of that message buffer should be equal or greater than the length of the NM vector configured by FLXAnFRNEMC.NML.**
2. **When the CC transits to HALT state, the cycle count is not incremented and therefore the NM vector is not updated. In this case FLXAnFRNMV1…3 holds the value from the cycle before.**

### 20.3.8 Filtering and Masking

Filtering is done by comparison of the configuration of assigned message buffers against actual slot and cycle counter values and channel ID (channel A, B). A message buffer is only updated / transmitted if the required matches occur.

Filtering is done on:

- Slot Counter
- Cycle Counter
- Channel ID

The following filter combinations for acceptance / transmit filtering are allowed:

- Slot Counter + Channel ID
- Slot Counter + Cycle Counter + Channel ID

All configured filters must match in order to store a received message in a message buffer.

**CAUTION**

**For the FIFO the acceptance filter is configured by the FIFO Rejection Filter (FLXAnFRFRF) and the FIFO Rejection Filter Mask (FLXAnFRFRFM).**

A message will be transmitted in the time slot corresponding to the configured frame ID on the configured channel(s). If cycle counter filtering is enabled the configured cycle filter value must also match.

#### 20.3.8.1 Slot Counter Filtering

Every transmit and receive buffer contains a frame ID stored in the header section. This frame ID is compared against the actual slot counter value in order to assign receive and transmit buffers to the corresponding slot.

If two or more message buffers are configured with the same frame ID and channel ID, and if they have a matching cycle counter filter value for the same slot, then the message buffer with the lowest message buffer number is used.

#### 20.3.8.2 Cycle Counter Filtering

Cycle counter filtering is based on the notion of a cycle set. For filtering purposes, a match is detected if any one of the elements of the cycle set is matched. The cycle set is defined by the cycle code field in header section 1 of each message buffer.

If message buffer 0 resp. 1 is configured to hold the startup / sync frame or the single slot frame by bits FLXAnFRSUCC1.TXST, FLXAnFRSUCC1.TXSY, and FLXAnFRSUCC1.TSM, cycle counter filtering for message buffer 0 resp. 1 shall be disabled.

**CAUTION**

**Sharing of a static time slot via cycle counter filtering between different nodes of a FlexRay network is not allowed.**

The set of cycle numbers belonging to a cycle set is determined as described.

**Table 20.112 Definition of cycle set**

| Cycle Code | Matching Cycle Counter Values |
|---|---|
| 0b000000x | all Cycles |
| 0b000001c | every second Cycle at (Cycle Count) mod2 = c |
| 0b00001cc | every fourth Cycle at (Cycle Count) mod4 = cc |
| 0b0001ccc | every eighth Cycle at (Cycle Count) mod8 = ccc |
| 0b001cccc | every sixteenth Cycle at (Cycle Count) mod16 = cccc |
| 0b01ccccc | every thirty-second Cycle at (Cycle Count) mod32 = ccccc |
| 0b1cccccc | every sixty-fourth Cycle at (Cycle Count) mod64 = cccccc |

The below table gives some examples for valid cycle sets to be used for cycle counter filtering.

**Table 20.113 Examples for valid cycle sets**

| Cycle Code | Matching Cycle Counter Values |
|---|---|
| 0b0000011 | 1-3-5-7-…. -63 |
| 0b0000100 | 0-4-8-12-…. -60 |
| 0b0001110 | 6-14-22-30-…. -62 |
| 0b0011000 | 8-24-40-56 |
| 0b0100011 | 3-35 |
| 0b1001001 | 9 |

The received message is stored only if the cycle counter value of the cycle during which the message is received matches an element of the receive buffer's cycle set. Channel ID and frame ID must also be met.

The content of a transmit buffer is transmitted on the configured channel(s) when an element of the cycle set matches the current cycle counter value. Channel ID and frame ID must also be met.

### 20.3.8.3 Channel ID Filtering

There is a 2-bit channel filtering field (CH) located in the header section of each message buffer in the Message RAM. It serves as a filter for receive buffers, and as a control field for transmit buffers.

**Table 20.114 Channel filtering configuration**

| CH[1:0] | **Transmit Buffer** transmit frame | **Receive Buffer store valid receive frame** |
|---|---|---|
| 00 | no transmission | ignore frame |
| 01 | on channel A | received on channel A |
| 10 | on channel B | received on channel B |
| 11 | on both channels (static segment only) | received on channel A or B (store first semantically valid frame, static segment only) |

The contents of a transmit buffer is transmitted on the channels specified in the channel filtering field when the slot counter filtering and cycle counter filtering criteria are also met. Only in static segment a transmit buffer may be set up for transmission on both channels (CH = "$11_B$").

Valid received frames are stored if they are received on the channels specified in the channel filtering field when the slot counter filtering and cycle counter filtering criteria are also met. Only in static segment a receive buffer may be setup for reception on both channels (CH = “11”).

**CAUTION**

**If a message buffer is configured for the dynamic segment and both bits of the channel filtering field are set to 1, no frames are transmitted resp. received frames are ignored (same function as CH = “00”).**

#### 20.3.8.4 FIFO Filtering

For FIFO filtering registers FLXAnFRFRF and FLXAnFRFRFM are used. The FIFO filter consists of channel filter FLXAnFRFRF.CH, frame ID filter FLXAnFRFRF.FID, and cycle counter filter FLXAnFRFRF.CYF. Registers FLXAnFRFRF and FLXAnFRFRFM can be configured in DEFAULT_CONFIG or CONFIG state only. The filter configuration in the header section of message buffers belonging to the FIFO is ignored.

The 7-bit cycle counter filter determines the cycle set to which frame ID and channel rejection filter are applied. In cycles not belonging to the cycle set specified by FLXAnFRFRF.CYF, all frames are rejected.

A valid received frame is stored in the FIFO if channel ID, frame ID, and cycle counter are not rejected by the configured rejection filter and rejection filter mask, and if there is no matching dedicated receive buffer.

## 20.3.9 Transmit Process

### 20.3.9.1 Static Segment

For the static segment, if there are several messages pending for transmission, the message with the frame ID corresponding to the next sending slot is selected for transmission.

The data section of transmit buffers assigned to the static segment can be updated until the end of the preceding time slot. This means that a transfer from the Input Buffer has to be started by writing to the Input Buffer Command Request register latest at this time.

### 20.3.9.2 Dynamic Segment

In the dynamic segment, if several messages are pending, the message with the highest priority (lowest frame ID) is selected next. In the dynamic segment different slot counter sequences on channel A and channel B are possible (concurrent sending of different frame IDs on both channels).

The data section of transmit buffers assigned to the dynamic segment can be updated until the end of the preceding slot. This means that a transfer from the Input Buffer has to be started by writing to the Input Buffer Command Request register latest at this time.

The start of latest transmit configured by FLXAnFRMHDC.SLT defines the maximum minislot value allowed before inhibiting new frame transmission in the dynamic segment of the current cycle.

### 20.3.9.3 Transmit Buffers

FlexRay message buffers can be configured as transmit buffers by programming bit CFG in the header section of the respective message buffer to 1 via FLXAnFRWRHS1.

There exist the following possibilities to assign a transmit buffer to the CC channels:

- Static segment: channel A or channel B,
  channel A and channel B
- Dynamic segment: channel A or channel B

Message buffer 0 resp. 1 is dedicated to hold the startup frame, the sync frame, or the designated single slot frame as configured by FLXAnFRSUCC1.TXST, FLXAnFRSUCC1.TXSY, and FLXAnFRSUCC1.TSM. In this case, it can be reconfigured in DEFAULT_CONFIG or CONFIG state only. This ensures that any node transmits at most one startup / sync frame per communication cycle. Transmission of startup / sync frames from other message buffers is not possible.

All other message buffers configured for transmission in static or dynamic segment are reconfigurable during runtime depending on the configuration of FLXAnFRMRC.SEC (see **Section 20.3.12.1, Reconfiguration of Message Buffers**). Due to the organization of the data partition in the Message RAM (reference by data pointer), reconfiguration of the configured payload length and the data pointer in the header section of a message buffer may lead to erroneous configurations.

If a message buffer is reconfigured (header section updated) during runtime, it may happen that this message buffer is not send out in the respective communication cycle.

The CC does not have the capability to calculate the header CRC. The Host is supposed to provide the header CRCs for all transmit buffers. If network management is required, the Host has to set the PPIT bit in the header section of the respective message buffer to 1 and write the network management information to the data section of the message buffer (see **Section 20.3.7, Network Management**).

The payload length field configures the payload length in 2-byte words. If the configured payload length of a static transmit buffer is shorter than the payload length configured for the static segment by

FLXAnFRMHDC.SFDL, the CC generates padding bytes to ensure that frames have proper physical length. The padding pattern is "$0000_H$".

**CAUTION**

**In case of an odd payload length (PLC = 1, 3, 5,...) the application has to write zero to the last 16 bit of the message buffers data section to ensure that the padding pattern is "$0000_H$".**

Each transmit buffer provides a transmission mode flag TXM that allows the Host to configure the transmission mode for the transmit buffer. If this bit is set, the transmitter operates in the single-shot mode. If this bit is cleared, the transmitter operates in the continuous mode.

In single-shot mode the CC resets the respective TXR flag to 0 after transmission has completed. Now the Host may update the transmit buffer.

In continuous mode, the CC does not reset the respective transmission request flag TXR to 0 after successful transmission. In this case a frame is sent out each time the filter criteria match. The TXR flag can be reset to 0 by the Host by writing the respective message buffer number to the FLXAnFRIBCR register while bit FLXAnFRIBCM.STXRH is set to 0.

If two or more transmit buffers meet the filter criteria simultaneously, the transmit buffer with the lowest message buffer number will be transmitted in the respective slot.

### 20.3.9.4 Frame Transmission

The following steps are required to prepare a message buffer for transmission:

- Configure the transmit buffer in the Message RAM via FLXAnFRWRHS1, FLXAnFRWRHS2, and FLXAnFRWRHS3
- Write the data section of the transmit buffer via FLXAnFRWRDSn
- Transfer the configuration and message data from Input Buffer to the Message RAM by writing the number of the target message buffer to register FLXAnFRIBCR
- If configured in register FLXAnFRIBCM, the transmission request flag TXR for the respective message buffer will be set as soon as the transfer has completed, and the message buffer is ready for transmission.
- Check whether the message buffer has been transmitted by checking the respective TXR bit (TXR = 0) in the FLXAnFRTRXQ1/2/3/4 registers (single-shot mode only).

After transmission has completed, the respective TXR flag in the FLXAnFRTXRQ1/2/3/4 register is reset to 0 (single-shot mode), and, if bit MBI in the header section of the message buffer is set to 1, flag FLXAnFRSIR.TXI is set to 1. If enabled, an interrupt is generated.

### 20.3.9.5 Null Frame Transmission

If in static segment the Host does not set the transmission request flag to 1 before transmit time, the CC transmits a null frame with the null frame indication bit set to 0 and the payload data set to zero.

In the following cases the CC transmits a null frame:

- If the message buffer with the lowest message buffer number matching the filter criteria does not have its transmission request flag set (TXR = 0) to 1.
- No transmit buffer configured for the slot has a cycle counter filter that matches the current cycle. In this case, no message buffer status FLXAnFRMBS is updated.

Null frames are not transmitted in the dynamic segment.

## 20.3.10 Receive Process

### 20.3.10.1 Dedicated Receive Buffers

A portion of the FlexRay message buffers can be configured as dedicated receive buffers by programming bit CFG in the header section of the respective message buffer to 0 via FLXAnFRWRHS1.

The following possibilities exist to assign a receive buffer to the CC channels:

- Static segment: channel A or channel B,
  channel A and channel B (the CC stores the first semantically valid frame)
- Dynamic segment: channel A or channel B

The CC transfers the payload data of valid received messages from the shift register of the FlexRay channel protocol controller (channel A or B) to the receive buffer with the matching filter configuration. A receive buffer stores all frame elements except the frame CRC.

All message buffers configured for reception in static or dynamic segment are reconfigurable during runtime depending on the configuration of FLXAnFRMRC.SEC (see **Section 20.3.12.1, Reconfiguration of Message Buffers**). If a message buffer is reconfigured (header section updated) during runtime it may happen that in the respective communication cycle a received message is lost.

If two or more receive buffers meet the filter criteria simultaneously, the receive buffer with the lowest message buffer number is updated with the received message.

### 20.3.10.2 Frame Reception

The following steps are required to prepare a dedicated message buffer for reception:

- Configure the receive buffer in the Message RAM via FLXAnFRWRHS1, FLXAnFRWRHS2, and FLXAnFRWRHS3
- Transfer the configuration from Input Buffer to the Message RAM by writing the number of the target message buffer to register FLXAnFRIBCR

Once these steps are performed, the message buffer functions as an active receive buffer and participates in the internal acceptance filtering process which takes place every time the CC receives a message. The first matching receive buffer is updated from the received message.

If a valid payload segment was stored in the data section of a message buffer, the respective ND flag in the FLXAnFRNDAT1/2/3/4 registers is set to 1, and, if bit MBI in the header section of that message buffer is set to 1, flag FLXAnFRSIR.RXI is set to 1. If enabled, an interrupt is generated.

In case that bit ND was already set to 1 when the Message Handler updates the message buffer, bit FLXAnFRMBS.MLST of the respective message buffer is set to 1 and the unprocessed message data is lost.

If no frame, a null frame, or a corrupted frame was received in a slot, the data section of the message buffer configured for this slot is not updated. In this case only the respective message buffer status FLXAnFRMBS is updated.

When the Message Handler changed the message buffer status FLXAnFRMBS in the header section of a message buffer, the respective MBC flag in the FLXAnFRMBSC1/2/3/4 registers is set to 1, and if bit MBI in the header section of that message buffer is set, flag FLXAnFRSIR.MBSI is set to 1. If enabled an interrupt is generated.

If the payload length of a received frame PLR is longer than the value programmed by PLC in the header section of the respective message buffer, the data field stored in the message buffer is truncated to that length.

To read a receive buffer from the Message RAM via the Output Buffer, proceed as described in **Section 20.3.12.2 (2), Data Transfer from Message RAM to Output Buffer**.

**CAUTION**

**The ND and MBC flags are automatically cleared by the Message Handler when the payload data and the header of a received message have been transferred to the Output Buffer, respectively.**

### 20.3.10.3 Null Frame Reception

The payload segment of a received null frame is not copied into the matching dedicated receive buffer. If a null frame has been received, only the message buffer status FLXAnFRMBS of the matching message buffer is updated from the received null frame. All bits in header 2 and 3 of the matching message buffer remain unchanged. They are updated from received data frames only.

When the Message Handler changed the message buffer status FLXAnFRMBS in the header section of a message buffer, the respective MBC flag in the FLXAnFRMBSC1/2/3/4 register is set to 1, and if bit MBI in the header section of that message buffer is set to 1, flag FLXAnFRSIR.MBSI is set to 1. If enabled, an interrupt is generated.

## 20.3.11 FIFO Function

### 20.3.11.1 Description

A portion of the message buffers can be configured as a cyclic First-In-First-Out (FIFO) buffer. The group of message buffers belonging to the FIFO is contiguous in the register map starting with the message buffer referenced by FLXAnFRMRC.FFB and ending with the message buffer referenced by FLXAnFRMRC.LCB. Up to 127 message buffers can be assigned to the FIFO.

Every valid incoming message not matching with any dedicated receive buffer but passing the programmable FIFO filter is stored into the FIFO. In this case frame ID, payload length, receive cycle count, and the message buffer status FLXAnFRMBS of the addressed FIFO message buffer are overwritten with frame ID, payload length, receive cycle count, and the status from the received frame. Bit FLXAnFRSIR.RFNE shows that the FIFO is not empty, bit FLXAnFRSIR.RFCL is set to 1 when the receive FIFO fill level FLXAnFRFSR.RFFL is equal or greater than the critical level as configured by FLXAnFRFCL.CL, bit FLXAnFREIR.RFO shows that a FIFO overrun has been detected. If enabled, interrupts are generated.

If null frames are not rejected by the FIFO rejection filter, the null frames will be treated like data frames when they are stored into the FIFO.

There are two index registers associated with the FIFO. The PUT Index Register (PIDX) is an index to the next available location in the FIFO. When a new message has been received it is written into the message buffer addressed by the PIDX register. The PIDX register is then incremented and addresses the next available message buffer. If the PIDX register is incremented past the highest numbered message buffer of the FIFO, the PIDX register is loaded with the number of the first (lowest numbered) message buffer in the FIFO chain. The GET Index Register (GIDX) is used to address the next message buffer of the FIFO to be read. The GIDX register is incremented after transfer of the contents of a message buffer belonging to the FIFO to the Output Buffer. The PUT Index Register and the GET Index Register are not accessible by the Host.

The FIFO is completely filled when the PUT index (PIDX) reaches the value of the GET index (GIDX). When the next message is written to the FIFO before the oldest message has been read, both PUT index and GET index are incremented and the new message overwrites the oldest message in the FIFO. This will set FIFO overrun flag FLXAnFREIR.RFO to 1.

A FIFO non empty status is detected when the PUT index (PIDX) differs from the GET index (GIDX). In this case flag FLXAnFRSIR.RFNE is set to 1. This indicates that there is at least one received message in the FIFO. The FIFO empty, FIFO not empty, and the FIFO overrun states are explained in **Figure 20.11** for a three message buffer FIFO.

The programmable FlexRay FIFO Rejection Filter (FLXAnFRFRF) defines a filter pattern for messages to be rejected. The FIFO filter consists of channel filter, frame ID filter, and cycle counter filter. If bit FLXAnFRFRF.RSS is set to 1, all messages received in the static segment are rejected by the FIFO. If bit FLXAnFRFRF.RNF is set to 1, received null frames are not stored in the FIFO.

The FlexRay FIFO Rejection Filter Mask (FLXAnFRFRFM) specifies which bits of the frame ID filter in the FIFO Rejection Filter register are marked 'don't care' for rejection filtering.

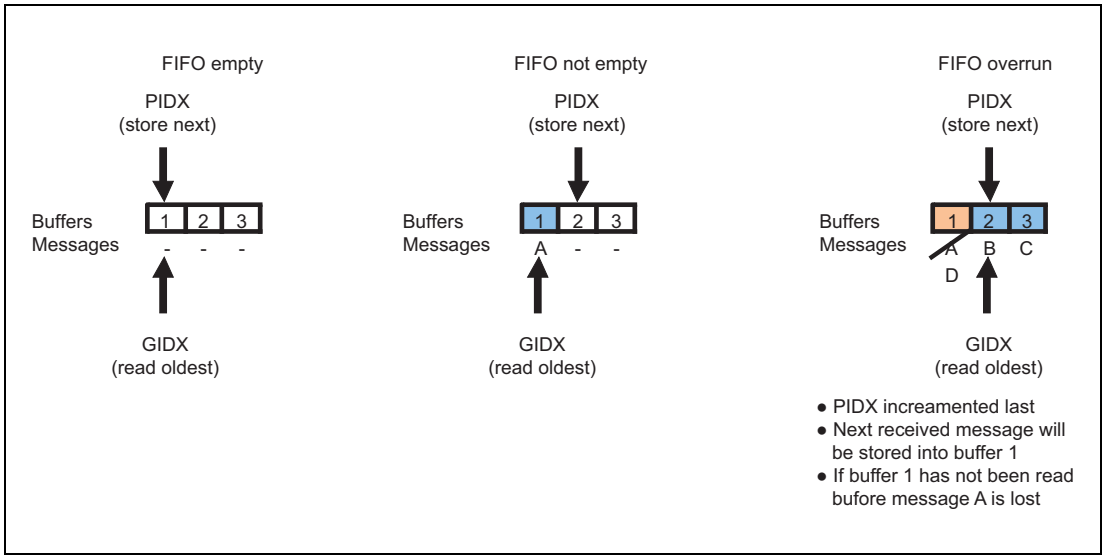



**Figure 20.11 FIFO Status: Empty, Not Empty, Overrun**

### 20.3.11.2 Configuration of the FIFO

(Re)configuration of message buffers belonging to the FIFO is only possible when the CC is in DEFAULT_CONFIG or CONFIG state. While the CC is in DEFAULT_CONFIG or CONFIG state, the FIFO function is not available.

For all message buffers belonging to the FIFO the payload length configured should be programmed to the same value via FLXAnFRWRHS2.PLC. The data pointer to the first 32-bit word of the data section of the respective message buffer in the Message RAM has to be configured via FLXAnFRWRHS3.DP.

All information required for acceptance filtering is taken from the FIFO rejection filter and the FIFO rejection filter mask. The values configured in the header sections of the message buffers belonging to the FIFO are, with exception of DP and PLC, irrelevant.

**CAUTIONS**

1. **It is recommended to program the MBI bits of the message buffers belonging to the FIFO to 0 via FLXAnFRWRHS1.MBI to avoid generation of RX interrupts.**
2. **If the payload length of a received frame is longer than the value programmed by FLXAnFRWRHS2.PLC in the header section of the respective message buffer, the data field stored in a message buffer of the FIFO is truncated to that length.**

#### 20.3.11.3 Access to the FIFO

**(1) When the output buffer is used:**

For FIFO access outside DEFAULT_CONFIG and CONFIG state, the Host has to trigger a transfer from the Message RAM to the Output Buffer by writing the number of the first message buffer of the FIFO (referenced by FLXAnFRMRC.FFB) to the register FLXAnFROBCR. The Message Handler then transfers the message buffer addressed by the GET Index Register (GIDX) to the Output Buffer. After this transfer the GET Index Register (GIDX) is incremented.

**(2) When the data transfer function is used:**

The message received in FIFO can be transferred to the Local RAM/Global RAM by using the output data transfer function. For the output data transfer function, see **Section 20.3.16.2, Output Data Transfer**, Output Data Transfer.

### 20.3.12 Message Handling

The Message Handler controls data transfers between the Input / Output Buffer and the Message RAM and between the Message RAM and the two Temporary buffers.

Access to the message buffers stored in the Message RAM is done under control of the Message Handler state machine. This avoids conflicts between accesses of the two FlexRay channel protocol controllers and the Host to the Message RAM.

Frame IDs of message buffers assigned to the static segment have to be in the range from 1 to FLXAnFRGTUC7.NSS. Frame IDs of message buffers assigned to the dynamic segment have to be in the range from FLXAnFRGTUC7.NSS + 1 to 2047.

Received messages with no matching dedicated receive buffer (static or dynamic segment) are stored in the receive FIFO (if configured) if they pass the FIFO rejection filter.

Access of the Host to the message buffer contents using the input or output buffer function is described in this subsection. Access to the message buffer contents using the data transfer function is mentioned in **Section 20.3.16, Usage of Data Transfer**.

#### 20.3.12.1 Reconfiguration of Message Buffers

In case that an application needs to operate with more than 128 different messages, static and dynamic message buffers may be reconfigured during FlexRay operation. This is done by updating the header section of the respective message buffer via Input Buffer registers FLXAnFRWRHS1…3.

Reconfiguration has to be enabled via control bits FLXAnFRMRC.SEC in the Message RAM Configuration register.

If a message buffer has not been transmitted / updated from a received frame before reconfiguration starts, the respective message is lost.

The point in time when a reconfigured message buffer is ready for transmission / reception according to the reconfigured frame ID depends on the actual state of the slot counter when the update of the header section has completed. Therefore it may happen that a reconfigured message buffer is not transmitted / updated from a received frame in the cycle where it was reconfigured.

The Message RAM is scanned according to below.

**Table 20.115 Scan of Message RAM**

| Start of Scan in Slot | Scan for Slots |
|---|---|
| 1 | 2…15, 1 (next cycle) |
| 8 | 16…23, 1 (next cycle) |
| 16 | 24…31, 1 (next cycle) |
| 24 | 32…39, 1 (next cycle) |
| … | … |

A Message RAM scan is terminated with the start of NIT regardless whether it has completed or not. The scan of the Message RAM for slots 2 to 15 starts at the beginning of slot 1 of the actual cycle. The scan of the Message RAM for slot 1 is done in the cycle before by checking in parallel to each scan of the Message RAM whether there is a message buffer configured for slot 1 of the next cycle.

The number of the first dynamic message buffer is configured by FLXAnFRMRC.FDB. In case a Message RAM scan starts while the CC is in dynamic segment, the scan starts with the message buffer number configured by FLXAnFRMRC.FDB.

In case a message buffer should be reconfigured to be used in slot 1 of the next cycle, the following has to be considered:

- If the message buffer to be reconfigured for slot 1 is part of the “Static Buffers”, it will only be found if it is reconfigured before the last Message RAM scan in the static segment of the actual cycle evaluates this message buffer.
- If the message buffer to be reconfigured for slot 1 is part of the “Static + Dynamic Buffers”, it will be found if it is reconfigured before the last Message RAM scan in the actual cycle evaluates this message buffer.
- The start of NIT terminates the Message RAM scan. In case the Message RAM scan has not evaluated the reconfigured message buffer until this point in time, the message buffer will not be considered for the next cycle.

**CAUTION**

**Reconfiguration of message buffers may lead to the loss of messages and therefore has to be used very carefully. In worst case (reconfiguration in consecutive cycles) it may happen that a message buffer is never transmitted / updated from a received frame.**

### 20.3.12.2 Host access to Message RAM

The message transfer between Input Buffer and Message RAM as well as between Message RAM and Output Buffer is triggered by the Host by writing the number of the target / source message buffer to be accessed to FLXAnFRIBCR or FLXAnFROBCR register.

The FLXAnFRIBCM and FLXAnFROBCM registers can be used to write / read header and data section of the selected message buffer separately.

If bit FLXAnFRIBCM.STXR is set to = 1, the transmission request flag TXR of the selected message buffer is automatically set to 1 after the message buffer has been updated. If bit FLXAnFRIBCM.STXR is reset to 0, the transmission request flag TXR of the selected message buffer is reset. This can be used to stop transmission from message buffers operated in continuous mode.

Input Buffer (IBF) and Output Buffer (OBF) are build up as a double buffer structure. One half of this double buffer structure is accessible by the Host (IBF Host / OBF Host), while the other half (IBF Shadow / OBF Shadow) is accessed by the Message Handler for data transfers between IBF / OBF and Message RAM.

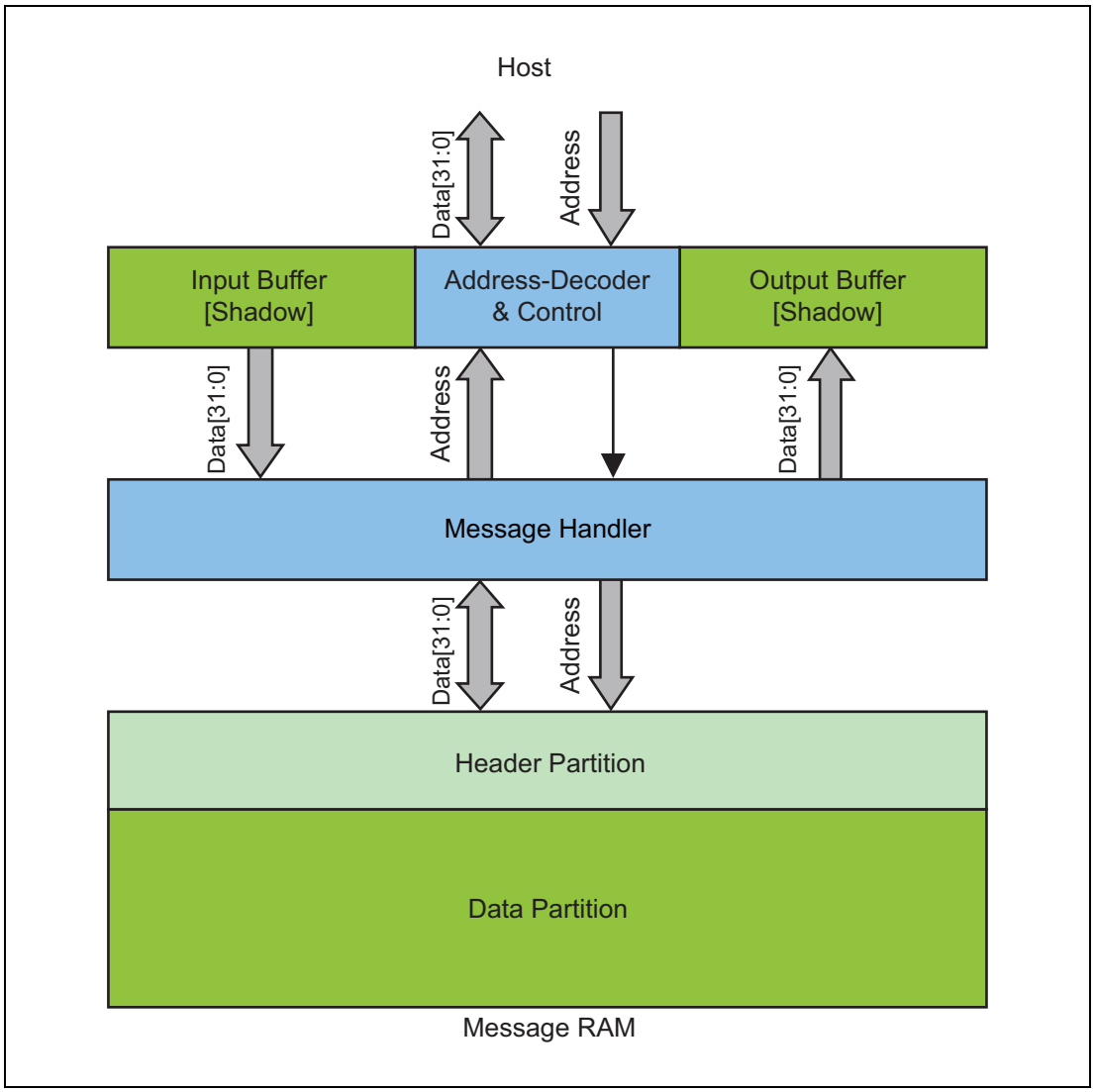



**Figure 20.12 Host access to Message RAM**

### (1) Data Transfer from Input Buffer to Message RAM

To configure / update a message buffer in the Message RAM, the Host has to write the data to FLXAnFRWRDSn and the header to FLXAnFRWRHS1…3. The specific action is selected by configuring the FlexRay Input Buffer Command Mask FLXAnFRIBCM.

When the Host writes the number of the target message buffer in the Message RAM to FLXAnFRIBCR.IBRH, IBF Host and IBF Shadow are swapped (see **Figure 20.14**).

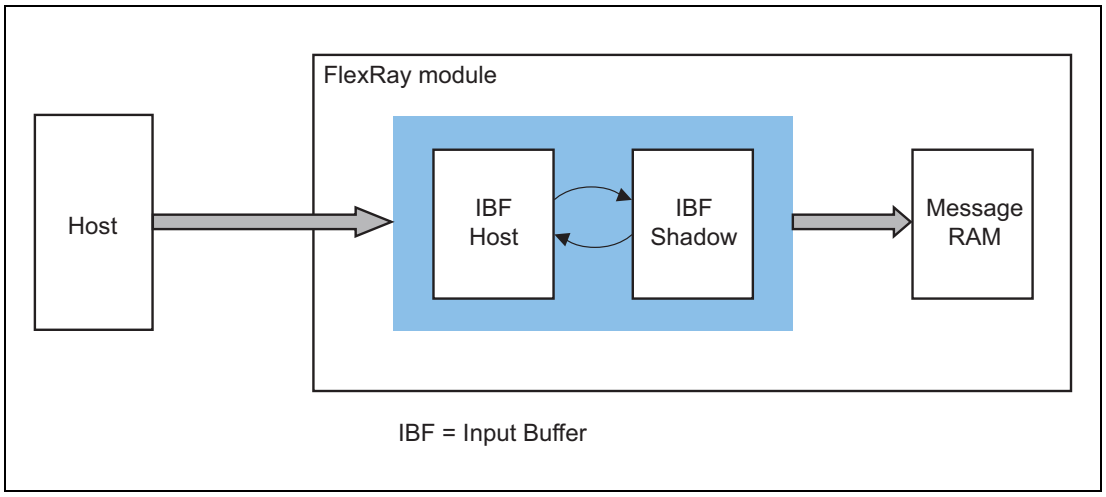



**Figure 20.13 Double Buffer Structure Input Buffer**

In addition the bits in the FLXAnFRIBCM and FLXAnFRIBCR registers are also swapped to keep them attached to the respective IBF section (see **Figure 20.14**).

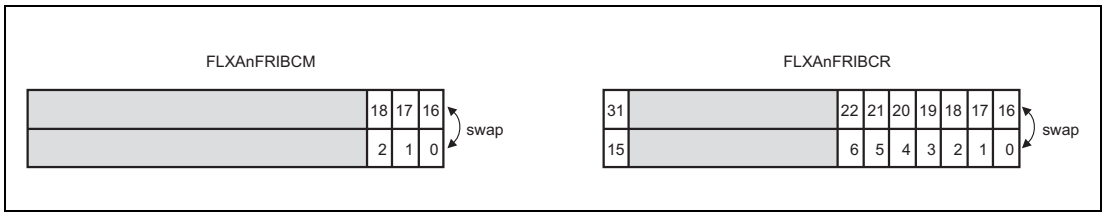



**Figure 20.14 Swapping of FLXAnFRIBCM and FLXAnFRIBCR bits**

With this write operation bit FLXAnFRIBCR.IBSYS is set to 1. The Message Handler then starts to transfer the contents of IBF Shadow to the message buffer in the Message RAM selected by FLXAnFRIBCR.IBRS.

While the Message Handler transfers the data from IBF Shadow to the target message buffer in the Message RAM, the Host may write the next message to IBF Host. After the transfer between IBF Shadow and the Message RAM has completed, bit FLXAnFRIBCR.IBSYS is set back to 0 and the next transfer to the Message RAM may be started by the Host by writing the respective target message buffer number to FLXAnFRIBCR.IBRH.

If a write access to FLXAnFRIBCR.IBRH occurs while FLXAnFRIBCR.IBSYS is 1, FLXAnFRIBCR.IBSYH is set to 1. After completion of the ongoing data transfer from IBF Shadow to the Message RAM, IBF Host and IBF Shadow are swapped, FLXAnFRIBCR.IBSYH is reset to 0, FLXAnFRIBCR.IBSYS remains set to 1, and the next transfer to the Message RAM is started. In addition the message buffer numbers stored under FLXAnFRIBCR.IBRH and FLXAnFRIBCR.IBRS and the command mask flags are also swapped.

**Example of a 8/16/32-bit Host access sequence:**

Configure / update n-th message buffer via IBF

- Wait until FLXAnFRIBCR.IBSYH is reset
- Write data section to FLXAnFRWRDSn
- Write header section to FLXAnFRWRHS1…3
- Write Command Mask: write FLXAnFRIBCM.STXRH, FLXAnFRIBCM.LDSH, FLXAnFRIBCM.LHSH
- Demand data transfer to target message buffer: write FLXAnFRIBCR.IBRH

Configure / update (n+1)th message buffer via IBF

- Wait until FLXAnFRIBCR.IBSYH is reset
- Write data section to FLXAnFRWRDSn
- Write header section to FLXAnFRWRHS1…3
- Write Command Mask: write FLXAnFRIBCM.STXRH, FLXAnFRIBCM.LDSH, FLXAnFRIBCM.LHSH
- Demand data transfer to target message buffer: write FLXAnFRIBCR.IBRH

**CAUTION**

**Any write access to IBF while FLXAnFRIBCR.IBSYH is 1 will set error flag FLXAnFREIR.IIBA to 1. In this case the write access has no effect.**

**Table 20.116 Assignment of FLXAnFRIBCM Bits**

| Pos. | Access | Bit | Function |
|---|---|---|---|
| 18 | r | STXRS | Set Transmission Request Shadow ongoing or finished |
| 17 | r | LDSS | Load Data Section Shadow ongoing or finished |
| 16 | r | LHSS | Load Header Section Shadow ongoing or finished |
| 2 | r/w | STXRH | Set Transmission Request Host |
| 1 | r/w | LDSH | Load Data Section Host |
| 0 | r/w | LHSH | Load Header Section Host |

**Table 20.117 Assignment of FLXAnFRIBCR Bits**

| Pos. | Access | Bit | Function |
|---|---|---|---|
| 31 | r | IBSYS | IBF Busy Shadow, signals ongoing transfer from IBF Shadow to Message RAM |
| 22…16 | r | IBRS | IBF Request Shadow, number of message buffer currently / lately updated |
| 15 | r | IBSYH | IBF Busy Host, transfer request pending for message buffer referenced by IBRH |
| 6…0 | r/w | IBRH | IBF Request Host, number of message buffer to be updated next |

### (2) Data Transfer from Message RAM to Output Buffer

To read a message buffer from the Message RAM, the Host has to write to register FLXAnFROBCR to trigger the data transfer as configured in FLXAnFROBCM. After the transfer has completed, the Host can read the transferred data from FLXAnFRRDDSn, FLXAnFRRDHS1…3, and FLXAnFRMBS.

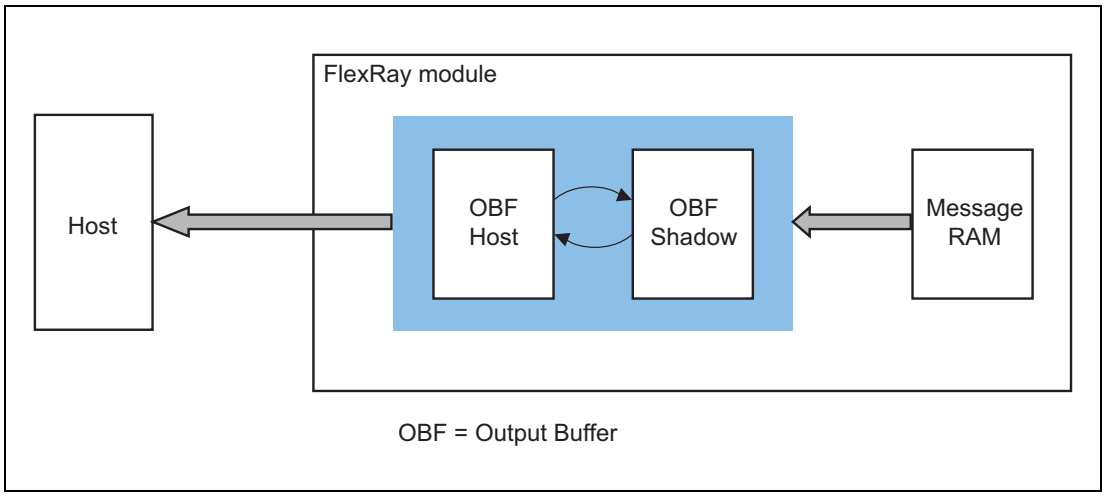



**Figure 20.15 Double buffer structure Output Buffer**

OBF Host and OBF Shadow as well as bits FLXAnFROBCM.RHSS, FLXAnFROBCM.RDSS, FLXAnFROBCM.RHSH, FLXAnFROBCM.RDSH and bits FLXAnFROBCR.OBRS, FLXAnFROBCR.OBRH are swapped under control of bits FLXAnFROBCR.VIEW and FLXAnFROBCR.REQ.

Writing bit FLXAnFROBCR.REQ to 1 copies bits FLXAnFROBCM.RHSS, FLXAnFROBCM.RDSS and bits FLXAnFROBCR.OBRS to an internal storage (see **Figure 20.16**).

After setting FLXAnFROBCR.REQ to 1, FLXAnFROBCR.OBSYS is set to 1, and the transfer of the message buffer selected by FLXAnFROBCR.OBRS from the Message RAM to OBF Shadow is started. After the transfer between the Message RAM and OBF Shadow has completed, the FLXAnFROBCR.OBSYS bit is set back to 0. Bits FLXAnFROBCR.REQ and FLXAnFROBCR.VIEW can only be set to 1 while FLXAnFROBCR.OBSYS is 0.

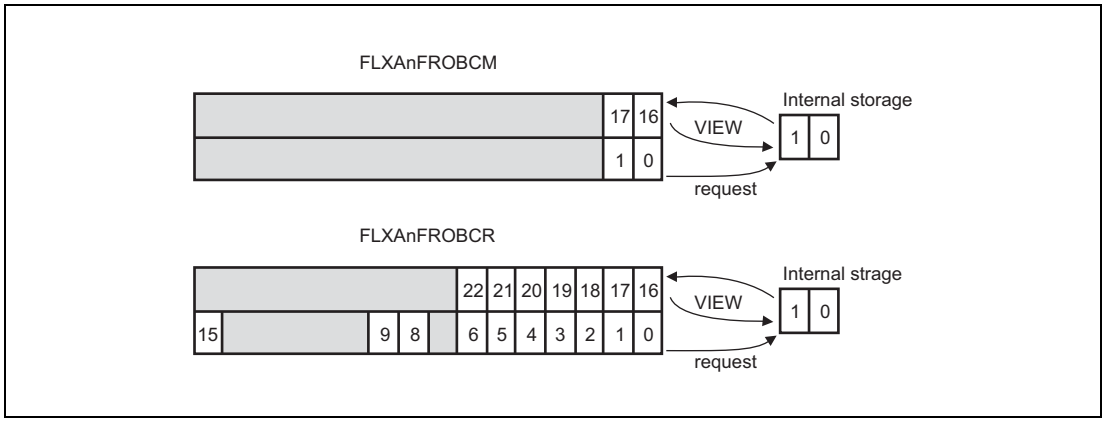



**Figure 20.16 Swapping of FLXAnFROBCM and FLXAnFROBCR bits**

OBF Host and OBF Shadow are swapped by setting bit FLXAnFROBCR.VIEW to 1 while bit FLXAnFROBCR.OBSYS is 0 (see **Figure 20.15**).

In addition bits FLXAnFROBCR.OBRH and bits FLXAnFROBCM.RHSH, FLXAnFROBCM.RDSH are swapped with the registers internal storage thus assuring that the message buffer number stored in FLXAnFROBCR.OBRH and the mask configuration stored in FLXAnFROBCM.RHSH, FLXAnFROBCM.RDSH matches the transferred data stored in OBF Host (see **Figure 20.16**).

Now the Host can read the transferred message buffer from OBF Host while the Message Handler may transfer the next message from the Message RAM to OBF Shadow.

If bits REQ and VIEW are set to 1 with the same write access while FLXAnFROBSYS is 0, FLXAnFROBSYS is automatically set to 1 and OBF Shadow and OBF Host are swapped. Additionally mask bits FLXAnFROBCM.RDSH and FLXAnFROBCM.RHSH are swapped with the registers internal storage to keep them attached to the respective Output Buffer transfer. Afterwards FLXAnFROBCR.OBRS is copied to the register internal storage, mask bits FLXAnFROBCM.RDSS and FLXAnFROBCM.RHSS are copied to register FLXAnFROBCM internal storage, and the transfer of the selected message buffer from the Message RAM to OBF Shadow is started. While the transfer is ongoing the Host can read the message buffer transferred by the previous transfer from OBF Host. When the current transfer between Message RAM and OBF Shadow has completed, this is signaled by setting FLXAnFROBCR.OBSYS back to 0.

#### Example of an 8/16/32-bit Host access to a single message buffer:

If a single message buffer has to be read out, two separate write accesses to FLXAnFROBCR.REQ and FLXAnFROBCR.VIEW are necessary:

- Wait until FLXAnFROBCR.OBSYS is reset
- Write Output Buffer Command Mask FLXAnFROBCM.RHSS, FLXAnFROBCM.RDSS
- Request transfer of message buffer to OBF Shadow by writing FLXAnFROBCR.OBRS and FLXAnFROBCR.REQ (in case of and 8-bit Host interface, FLXAnFROBCR.OBRS has to be written before FLXAnFROBCR.REQ).
- Wait until FLXAnFROBCR.OBSYS is reset
- Toggle OBF Shadow and OBF Host by writing FLXAnFROBCR.VIEW = 1
- Read out transferred message buffer by reading FLXAnFRRDDSn, FLXAnFRRDHS1…3, and FLXAnFRMBS

#### Example of an 8/16/32-bit Host access sequence:

Request transfer of 1st message buffer to OBF Shadow

- Wait until FLXAnFROBCR.OBSYS is reset
- Write Output Buffer Command Mask FLXAnFROBCM.RHSS, FLXAnFROBCM.RDSS for 1st message buffer
- Request transfer of 1st message buffer to OBF Shadow by writing FLXAnFROBCR.OBRS and FLXAnFROBCR.REQ (in case of an 8-bit Host interface, FLXAnFROBCR.OBRS has to be written before FLXAnFROBCR.REQ).

Toggle OBF Shadow and OBF Host to read out 1st transferred message buffer and request transfer of 2nd message buffer:

- Wait until FLXAnFROBCR.OBSYS is reset to 0

- Write Output Buffer Command Mask FLXAnFROBCM.RHSS, FLXAnFROBCM.RDSS for 2nd message buffer
- Toggle OBF Shadow and OBF Host and start transfer of 2nd message buffer to OBF Shadow simultaneously by writing FLXAnFROBCR.OBRS of 2nd message buffer, FLXAnFROBCR.REQ, and FLXAnFROBCR.VIEW (in case of and 8-bit Host interface, FLXAnFROBCR.OBRS has to be written before FLXAnFROBCR.REQ and FLXAnFROBCR.VIEW).
- Read out 1st transferred message buffer by reading FLXAnFRRDDSn, FLXAnFRRDHS1…3, and FLXAnFRMBS

…

Demand access to last requested message buffer without request of another message buffer:

- Wait until FLXAnFROBCR.OBSYS is reset to 0
- Demand access to last transferred message buffer by writing FLXAnFROBCR.VIEW
- Read out last transferred message buffer by reading FLXAnFRRDDSn, FLXAnFRRDHS1…3, and FLXAnFRMBS

**Table 20.118 Assignment of FLXAnFROBCM bits**

| Pos. | Access | Bit | Function |
|---|---|---|---|
| 17 | r | RDSH | Data Section available for Host access |
| 16 | r | RHSH | Header Section available for Host access |
| 1 | r/w | RDSS | Read Data Section Shadow |
| 0 | r/w | RHSS | Read Header Section Shadow |

**Table 20.119 Assignment of FLXAnFROBCR bits**

| Pos. | Access | Bit | Function |
|---|---|---|---|
| 22…16 | r | OBRH | OBF Request Host,<br>number of message buffer available for Host access |
| 15 | r | OBSYS | OBF Busy Shadow,<br>signals ongoing transfer from Message RAM to OBF Shadow |
| 9 | r/w | REQ | Request Transfer from Message RAM to OBF Shadow |
| 8 | r/w | VIEW | View OBF Shadow, swap OBF Shadow and OBF Host |
| 6…0 | r/w | OBRS | OBF Request Shadow, number of message buffer for next request |

### 20.3.12.3 FlexRay Protocol Controller Access to Message RAM

The two Temporary buffers (TBF A, B) are used to buffer the data for transfer between the two FlexRay Protocol Controllers and the Message RAM.

Each Temporary buffer is build up as a double buffer, able to store two complete FlexRay messages. There is always one buffer assigned to the corresponding Protocol Controller while the other one is accessible by the Message Handler.

If e.g. the Message Handler writes the next message to be sent to Temporary buffer Tx, the FlexRay Channel Protocol Controller can access Temporary buffer Rx to store the message it is actually receiving. During transmission of the message stored in Temporary buffer Tx, the Message Handler transfers the last received message stored in Temporary buffer Rx to the Message RAM (if it passes acceptance filtering) and updates the respective message buffer.

Data transfers between the Temporary buffers and the shift registers of the FlexRay Channel Protocol Controllers are done in words of 32 bit. This enables the use of a 32 bit shift register independent of the length of the FlexRay messages.

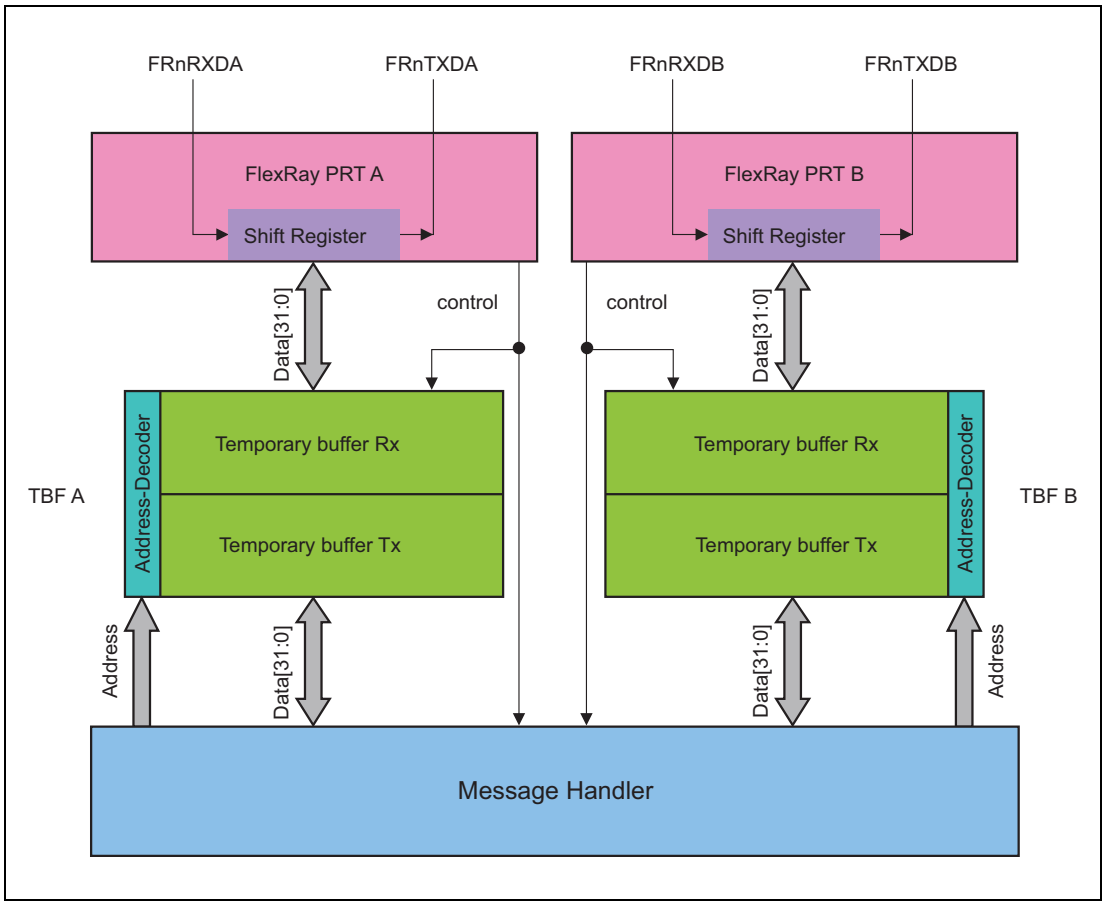



**Figure 20.17 Access to Temporary Buffers**

### 20.3.13 Message RAM

To avoid conflicts between Host access to the Message RAM and FlexRay message reception / transmission, the Host cannot directly access the message buffers in the Message RAM. These accesses are handled via the Input and Output Buffers. The Message RAM is able to store up to 128 message buffers depending on the configured payload length.

The Message RAM is able to store up to 2048 32-bit words. To achieve the required flexibility with respect to different numbers of data bytes per FlexRay frame (0…254), the Message RAM has a structure as shown in **Figure 20.18**.

The data partition is allowed to start at Message RAM word number: (FLXAnFRMRC.LCB + 1) * 4

When the Message Buffer which Data Section is located directly following the Header Partition is configured as receive buffer(FLXAnFRWRHS1.CFG set to "0") or receive FIFO, the user should leave at least 32bit unused area at the head of Data Partition. In this case the data partition is allowed to start at Message RAM word number: ((bits FLXAnFRMRC.LCB[7:0] + 1) * 4) + 1.

When the Message Buffer which Data Section is located directly following the Header Partition is configured as transmit buffer (FLXAnFRWRHS1.CFG set to "1"), the data partition is allowed to start at Message RAM word number:(bits FLXAnFRMRC.LCB[7:0] + 1) * 4.

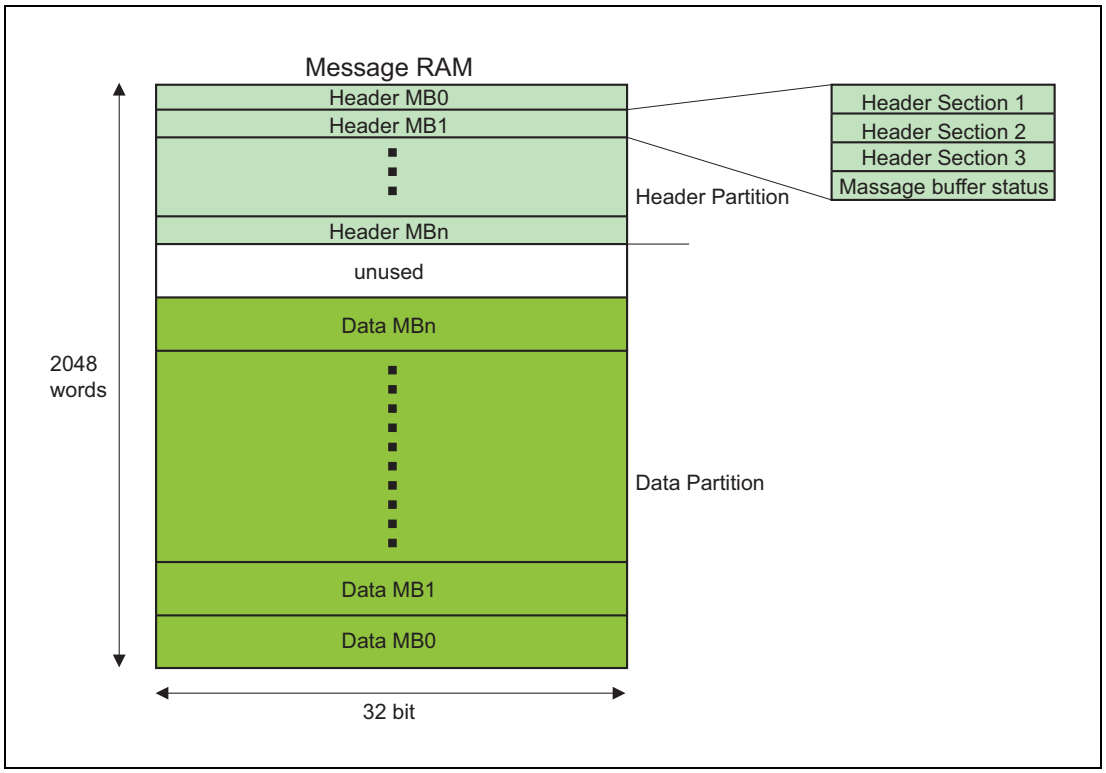



**Figure 20.18 Configuration Example of Message Buffers in the Message RAM**

#### Header Partition

Stores header sections of the configured message buffers:

- Supports a maximum of 128 message buffers
- Each message buffer has a header section of four 32 bit words
- Header 3 of each message buffer holds the 11-bit data pointer to the respective data section in the data partition

**Data Partition**

Flexible storage of data sections with different length. Some maximum values are:

- 30 message buffers with 254 byte data section each
- Or 56 message buffers with 128 byte data section each
- Or 128 message buffers with 48 byte data section each

**CAUTION**

**header partition + data partition may not occupy more than 2048 32-bit words.**

### 20.3.13.1 Header Partition

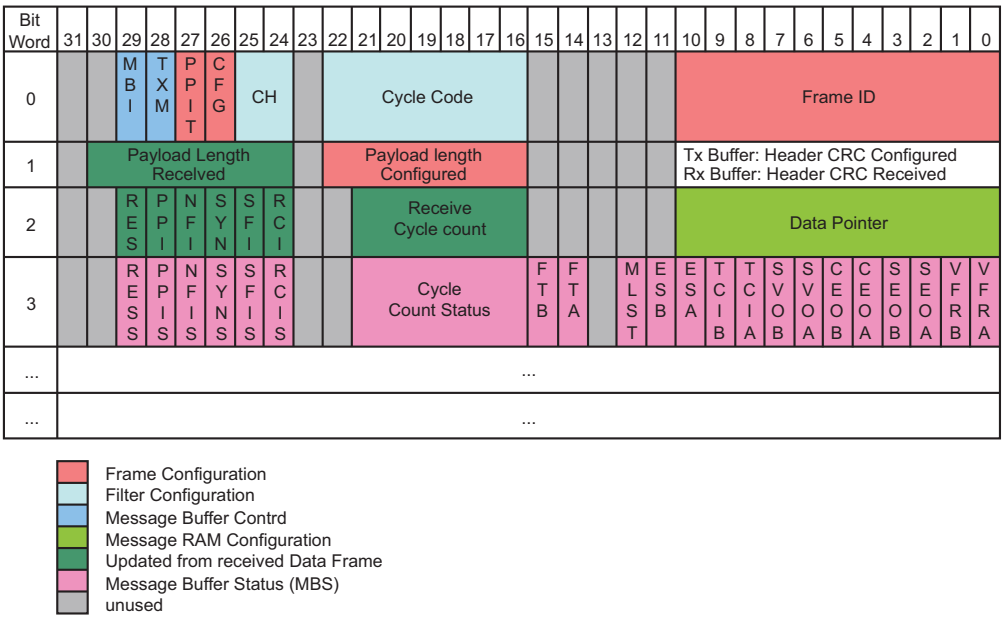
The elements used for configuration of a message buffer as well as the actual message buffer status are stored in the header partition of the Message RAM as listed in below. Configuration of the header sections of the message buffers is done via IBF (FLXAnFRWRHS1…3). Read access to the header sections is done via OBF (FLXAnFRRDHS1…3 + FLXAnFRMBS). The data pointer has to be calculated by the programmer to define the starting point of the data section for the respective message buffer in the data partition of the Message RAM. The data pointer should not be modified during runtime. For message buffers belonging to the receive FIFO (re)configuration is possible in DEFAULT_CONFIG or CONFIG state only.

The header section of each message buffer occupies four 32-bit words in the header partition of the Message RAM. The header of message buffer 0 starts with the first word in the Message RAM.

For transmit buffers the Header CRC has to be calculated by the Host.

Payload Length Received PLR, Receive Cycle Count RCC, Received on Channel Indicator RCI, Startup Frame Indicator SFI, Sync Frame Indicator SYN, Null Frame Indicator NFI, Payload Preamble Indicator PPI, and Reserved Bit RES are updated from received valid data frames only.

**Table 20.120 Header Section of a Message Buffer in the Message RAM**



**(1) Header section 1 (word 0)**

Write access via FLXAnFRWRHS1, read access via FLXAnFRRDHS1:

- Frame ID
  - Slot counter filtering configuration
- Cycle Code
  - Cycle counter filtering configuration
- CH
  - Channel filtering configuration
- CFG
  - Message buffer direction configuration: receive / transmit
- PPIT
  - Payload Preamble Indicator Transmit
- TXM
  - Transmit mode configuration: single-shot / continuous
- MBI
  - Message buffer receive / transmit interrupt enable

**(2) Header section 2 (word 1)**

Write access via FLXAnFRWRHS2, read access via FLXAnFRRDHS2:

- Header CRC
  - Transmit Buffer: Configured by the Host (calculated from frame header)
  - Receive Buffer: Updated from received frame
- Payload Length Configured
  - Length of data section (2-byte words) as configured by the Host
- Payload Length Received
  - Length of payload segment (2-byte words) stored from received frame

**(3) Header section 3 (word 2)**

Write access via FLXAnFRWRHS3, read access via FLXAnFRRDHS3:

- Data Pointer
  - Pointer to the beginning of the corresponding data section in the data partition

Read access via FLXAnFRRDHS3, valid for receive buffers only, updated from received frames:

- Receive Cycle Count
  - Cycle count from received frame
- RCI
  - Received on Channel Indicator

- SFI
  - Startup Frame Indicator
- SYN
  - Sync Frame Indicator
- NFI
  - Null Frame Indicator
- PPI
  - Payload Preamble Indicator
- RES
  - Reserved bit

#### (4) Message Buffer Status FLXAnFRMBS (word 3)

Read access via FLXAnFRMBS, updated by the CC at the end of the configured slot.

- VFRA
  - Valid Frame Received on channel A
- VFRB
  - Valid Frame Received on channel B
- SEOA
  - Syntax Error Observed on channel A
- SEOB
  - Syntax Error Observed on channel B
- CEOA
  - Content Error Observed on channel A
- CEOB
  - Content Error Observed on channel B
- SVOA
  - Slot boundary Violation Observed on channel A
- SVOB
  - Slot boundary Violation Observed on channel B
- TCIA
  - Transmission Conflict Indication channel A
- TCIB
  - Transmission Conflict Indication channel B
- ESA
  - Empty Slot Channel A

- ESB
  - Empty Slot Channel B
- MLST
  - Message LoST
- FTA
  - Frame Transmitted on Channel A
- FTB
  - Frame Transmitted on Channel B
- Cycle Count Status
  - Actual cycle count when status was updated
- RCIS
  - Received on Channel Indicator Status
- SFIS
  - Startup Frame Indicator Status
- SYNS
  - Sync Frame Indicator Status
- NFIS
  - Null Frame Indicator Status
- PPIS
  - Payload Preamble Indicator Status
- RESS
  - Reserved bit Status

### 20.3.13.2 Data Partition

The data partition of the Message RAM stores the data sections of the message buffers configured for reception / transmission as defined in the header partition. The number of data bytes for each message buffer can vary from 0 to 254. To optimize the data transfer between the shift registers of the two FlexRay Protocol Controllers and the Message RAM as well as between the Host interface and the Message RAM, the physical width of the Message RAM is set to 4 bytes.

The data partition starts after the last word of the header partition. When configuring the message buffers in the Message RAM the programmer has to assure that the data pointers point to addresses within the data partition. **Table 20.121** below shows an example how the data sections of the configured message buffers can be stored in the data partition of the Message RAM.

The beginning and the end of a message buffer's data section is determined by the data pointer and the payload length configured in the message buffer's header section, respectively. This enables a flexible usage of the available RAM space for storage of message buffers with different data length.

If the size of the data section is an odd number of 2-byte words, the remaining 16 bits in the last 32bit word are unused (see **Table 20.121** below).

Table 20.121 Example for Structure of the Data Partition in the Message RAM

| Bit Word | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ... | unused | | | | | | | | unused | | | | | | | | unused | | | | | | | | unused | | | | | | | |
| ... | unused | | | | | | | | unused | | | | | | | | unused | | | | | | | | unused | | | | | | | |
| ... | MBn Data3 | | | | | | | | MBn Data2 | | | | | | | | MBn Data1 | | | | | | | | MBn Data0 | | | | | | | |
| ... | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | |
| ... | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | |
| ... | MBn Data(m) | | | | | | | | MBn Data(m-1) | | | | | | | | MBn Data(m-2) | | | | | | | | MBn Data(m-3) | | | | | | | |
| ... | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | |
| ... | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | |
| ... | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | |
| ... | MB1 Data3 | | | | | | | | MB1 Data2 | | | | | | | | MB1 Data1 | | | | | | | | MB1 Data0 | | | | | | | |
| ... | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | |
| 2046 | MB0 Data3 | | | | | | | | MB0 Data2 | | | | | | | | MB0 Data1 | | | | | | | | MB0 Data0 | | | | | | | |
| 2047 | unused | | | | | | | | unused | | | | | | | | MB0 Data5 | | | | | | | | MB0 Data4 | | | | | | | |

### 20.3.13.3 Message Data Integrity Check

There is a data integrity checking mechanism implemented in the FlexRay core to assure the integrity of the data stored in the related RAM. Each RAM has a checksum generator / checker attached as shown in **Figure 20.19**.

When data is written to a RAM, the local checksum generator generates the checksum. The checksum is stored together with the respective data word. The checksum is checked each time a data word is read from a RAM.

If a checksum error is detected, the respective access error flag is set. The access error flags FLXAnFRMHDS.AMR, FLXAnFRMHDS.ATBF1, FLXAnFRMHDS.ATBF2 and the faulty message buffer indicators FLXAnFRMHDS.FMBD, FLXAnFRMHDS.MFMB, FLXAnFRMHDS.FMB are located in the FlexRay Message Handler Status register. These single access error flags control the error interrupt flag FLXAnFREIR.AERR.

**Figure 20.19** shows the data paths between the Input Buffer, Temporary Buffer and Message RAM.

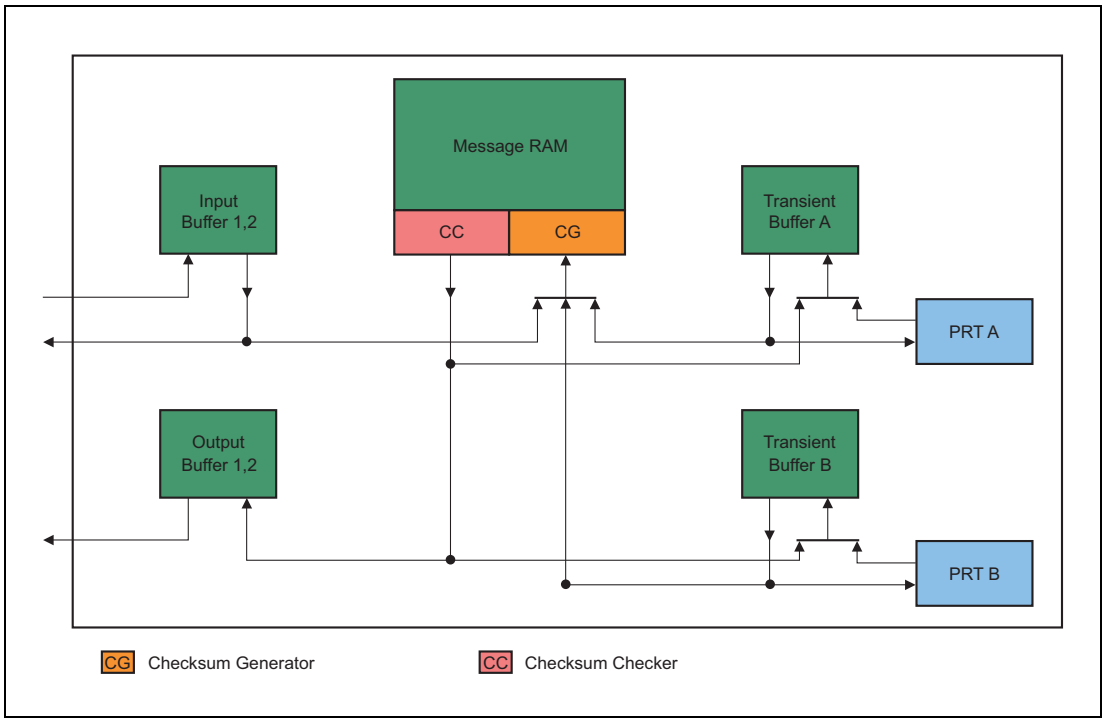



Figure 20.19 Checksum Generation and Check

When an access error has been detected the following actions will be performed:

**In all cases:**

- The respective access error flag in FLXAnFRMHDS register is set
- The access error flag FLXAnFREIR.AERR is set and, if enabled, a module interrupt to the Host will be generated.

**Additionally in specific cases:**

(1) Access error during data transfer from Input Buffer 1, 2 to Message RAM when reading header section of respective message buffer from Message RAM:

- FLXAnFRMHDS.AMR is set.
- FLXAnFRMHDS.FMBD bit is set to indicate that FLXAnFRMHDS.FMB points to a faulty message buffer.
- FLXAnFRMHDS.FMB indicates the number of the faulty message buffer.
- The data section of the respective message buffer is not updated.
- Transmit buffer: Transmission request for the respective message buffer is not set.

(2) Access error during scan of header sections in Message RAM:

- FLXAnFRMHDS.AMR is set.
- FLXAnFRMHDS.FMBD bit is set to indicate that FLXAnFRMHDS.FMB points to a faulty message buffer.
- FLXAnFRMHDS.FMB indicates the number of the faulty message buffer.
- Ignore message buffer (message buffer is skipped).

(3) Access error during data transfer from Message RAM to Temporary Buffer 1, 2:

- FLXAnFRMHDS.AMR is set.
- FLXAnFRMHDS.FMBD bit is set to indicate that FLXAnFRMHDS.FMB points to a faulty message buffer.
- FLXAnFRMHDS.FMB indicates the number of the faulty message buffer.
- Frame not transmitted, frames already in transmission are invalidated by setting the frame CRC to zero.

(4) Access error during data transfer from Temporary Buffer 1, 2 to Message RAM when reading header section of respective message buffer from Message RAM:

- FLXAnFRMHDS.AMR is set.
- FLXAnFRMHDS.FMBD bit is set to indicate that FLXAnFRMHDS.FMB points to a faulty message buffer.
- FLXAnFRMHDS.FMB indicates the number of the faulty message buffer.
- The data section of the respective message buffer is not updated.

(5) Access error during data transfer from Message RAM to Output Buffer:

- The access error flag FLXAnFRMHDS.AMR is set.

- FLXAnFRMHDS.FMBD bit is set to indicate that FLXAnFRMHDS.FMB points to a faulty message buffer.
- FLXAnFRMHDS.FMB indicates the number of the faulty message buffer.

(6) Access error during a data transfer from Temporary Buffer 1, 2 to Protocol Controller 1, 2:

- FLXAnFRMHDS.ATBF1, 2 bit is set.
- Frames already in transmission are invalidated by setting the frame CRC to zero.

(7) Access error during data transfer from Temporary Buffer 1, 2 to Message RAM when reading Temporary Buffer RAM 1, 2:

- FLXAnFRMHDS.ATBF1, 2 bit is set.
- FLXAnFRMHDS.FMBD bit is set to indicate that FLXAnFRMHDS.FMB points to a faulty message buffer.
- FLXAnFRMHDS.FMB indicates the number of the faulty message buffer.

(8) Access error during data read of Temporary Buffer RAM 1, 2:

- When an access error occurs while the Message Handler read a frame with network management information (PPI = 1) from the Temporary Buffer RAM 1, 2, the corresponding network management vector registers FLXAnFRNMV1 to 3 are not updated from this frame.

### 20.3.13.4 Host Handling of Access Errors

Access error caused by temporary bit flips can be fixed by:

#### (1) Self-healing

Access errors located in the Data Section of Message RAM, Temporary Buffer RAM A or Temporary Buffer RAM B are overwritten with the next write access to the disturbed bit(s) caused by Host access or by FlexRay communication.

#### (2) CLEAR_RAMS Command

The POC command CLEAR_RAMS initializes the message RAM to zero, when called in the DEFAULT_CONFIG or CONFIG state.

#### (3) Temporary Unlocking of Header Section

An access error in the header section of a locked message buffer can be fixed by a transfer from the Input Buffer to the locked buffer Header Section. For this transfer, the write access to the FLXAnFRIBCR (specifying the message buffer number) must be immediately preceded by the unlock sequence normally used to leave CONFIG state (see **Section 20.2.3.1, FLXAnFRLCK — FlexRay Lock Register**).

For that single transfer the respective message buffer header is unlocked, regardless whether it belongs to the FIFO or whether its locking is controlled by FLXAnFRMRC.SEC, and will be updated with new data.

### 20.3.14 Module Interrupt

In general, interrupts provide a close link to the protocol timing as they are triggered almost immediately when an error or status change is detected by the CC, a frame is received or transmitted, a configured timer interrupt is activated, or a stop watch event occurred. This enables the Host to react very quickly on specific error conditions, status changes, or timer events. On the other hand too many interrupts can cause the Host to miss deadlines required for the application. Therefore the CC supports enable / disable controls for each individual interrupt source separately.

An interrupt may be triggered when

- An error was detected
- A status flag is set to 1
- A timer reaches a preconfigured value
- A message transfer from Input Buffer to Message RAM or from Message RAM to Output Buffer has completed
- A message transfer from the Local RAM/Global RAM to Message RAM or from Message RAM to Local RAM/Global RAM has completed
- A stop watch event occurred

Tracking status and generating interrupts when a status change or an error occurs are two independent tasks. Regardless of whether an interrupt is enabled or not, the corresponding status is tracked and indicated by the CC. The Host has access to the actual status and error information by reading registers FLXAnFREIR, FLXAnFRSIR, FLXAnFROS, FLXAnFROTS and FLXAnFRITS.

The general purpose interrupt lines to the Host, FlexRay Interrupt 0, FlexRay Interrupt 1, are controlled by the enabled interrupts in FLXAnFREIES and FLXAnFRSIES. In addition each of the two interrupt lines can be enabled / disabled separately by programming bit FLXAnFRILE.EINT0 and FLXAnFRILE.EINT1.

The input data transfer interrupt lines to the Host, FlexRay input queue empty interrupt, FlexRay input queue full interrupt, are controlled by the enabled interrupts in FLXAnFRITS. In addition each of the input data transfer interrupts can be enabled / disabled separately by programming the related bits in FLXAnFRITC.

The output data transfer interrupt lines to the Host, FlexRay FIFO transfer warning interrupt, FlexRay output transfer warning interrupt, FlexRay FIFO transfer interrupt, FlexRay output transfer interrupt, are controlled by the enabled interrupts in FLXAnFROTS. In addition each of the output data transfer interrupts can be enabled / disabled separately by programming the related bits in FLXAnFROTC.

The three timer interrupts lines to the Host are controlled by the enabled interrupts in FLXAnFROS. In addition each of the interrupt lines can be enabled / disabled separately by programming bit FLXAnFROC.T0IE, FLXAnFROC.T1IE and FLXAnFROC.T2IE.

When a transfer between IBF / OBF and the Message RAM has completed bit FLXAnFRSIR.TIBC or FLXAnFRSIR.TOBC is set to 1.

### 20.3.15 Assignment of FlexRay Configuration Parameters

Table 20.122 FlexRay configuration parameters (1/2)

| Parameter | Bit (field) |
|---|---|
| pKeySlotusedForStartup | FLXAnFRSUCC1.TXST |
| pKeySlotUsedForSync | FLXAnFRSUCC1.TXSY |
| gColdStartAttempts | FLXAnFRSUCC1.CSA |
| pAllowPassiveToActive | FLXAnFRSUCC1.PTA |
| pWakeupChannel | FLXAnFRSUCC1.WUCS |
| pSingleSlotEnabled | FLXAnFRSUCC1.TSM |
| pAllowHaltDueToClock | FLXAnFRSUCC1.HCSE |
| pChannels | FLXAnFRSUCC1.CCH |
| pdListenTimeOut | FLXAnFRSUCC2.LT |
| gListenNoise | FLXAnFRSUCC2.LTN |
| gMaxWithoutClockCorrectionPassive | FLXAnFRSUCC3.WCP |
| gMaxWithoutClockCorrectionFatal | FLXAnFRSUCC3.WCF |
| gNetworkManagementVectorLength | FLXAnFRNEMC.NML |
| gdTSSTransmitter | FLXAnFRPRTC1.TSST |
| gdCASRxLowMax | FLXAnFRPRTC1.CASM |
| gdSampleClockPeriod | FLXAnFRPRTC1.BRP |
| pSamplesPerMicrotick | FLXAnFRPRTC1.BRP |
| gdWakeupSymbolRxWindow | FLXAnFRPRTC1.RXW |
| pWakeupPattern | FLXAnFRPRTC1.RWP |
| gdWakeupSymbolRxIdle | FLXAnFRPRTC2.RXI |
| gdWakeupSymbolRxLow | FLXAnFRPRTC2.RXL |
| gdWakeupSymbolTxIdle | FLXAnFRPRTC2.TXI |
| gdWakeupSymbolTxLow | FLXAnFRPRTC2.TXL |
| gPayloadLengthStatic | FLXAnFRMHDC.SFDL |
| pLatestTx | FLXAnFRMHDC.SLT |
| pMicroPerCycle | FLXAnFRGTUC1.UT |
| gMacroPerCycle | FLXAnFRGTUC2.MPC |
| gSyncNodeMax | FLXAnFRGTUC2.SNM |
| pMicroInitialOffset[A] | FLXAnFRGTUC3.UIOA |
| pMicroInitialOffset[B] | FLXAnFRGTUC3.UIOB |
| pMacroInitialOffset[A] | FLXAnFRGTUC3.MIOA |
| pMacroInitialOffset[B] | FLXAnFRGTUC3.MIOB |
| gdNIT | FLXAnFRGTUC4.NIT |
| gOffsetCorrectionStart | FLXAnFRGTUC4.OCS |
| pDelayCompensation[A] | FLXAnFRGTUC5.DCA |
| pDelayCompensation[B] | FLXAnFRGTUC5.DCB |
| pClusterDriftDamping | FLXAnFRGTUC5.CDD |
| pDecodingCorrection | FLXAnFRGTUC5.DEC |
| pdAcceptedStartupRange | FLXAnFRGTUC6.ASR |
| pdMaxDrift | FLXAnFRGTUC6.MOD |
| gdStaticSlot | FLXAnFRGTUC7.SSL |
| gNumberOfStaticSlots | FLXAnFRGTUC7.NSS |

**Table 20.122 FlexRay configuration parameters (2/2)**

| Parameter | Bit (field) |
|---|---|
| gdMinislot | FLXAnFRGTUC8.MSL |
| gNumberOfMinislots | FLXAnFRGTUC8.NMS |
| gdActionPointOffset | FLXAnFRGTUC9.APO |
| gdMinislotActionPointOffset | FLXAnFRGTUC9.MAPO |
| gdDynamicSlotIdlePhase | FLXAnFRGTUC9.DSI |
| pOffsetCorrectionOut | FLXAnFRGTUC10.MOC |
| pRateCorrectionOut | FLXAnFRGTUC10.MRC |
| pExternOffsetCorrection | FLXAnFRGTUC11.EOC |
| pExternRateCorrection | FLXAnFRGTUC11.ERC |

### 20.3.16 Usage of Data Transfer

A mechanism is implemented to allow storage of FlexRay messages directly into the Local RAM/ Global RAM (user RAM) and have transfers between the FlexRay internal message RAM and the Local RAM/Global RAM and vice versa with minimum CPU support. The data in the Local RAM/ Global RAM should be indexed by data structure pointers located in data pointer tables stored in the Local RAM/Global RAM.

Data transfer from the Local RAM/Global RAM to the FlexRay internal message RAM (input transfer) needs to be initiated by the application. These transfers can be used to configure message buffers or to update transmit data.

A data transfer from the FlexRay internal message RAM to the Local RAM/Global RAM (output transfer) is initiated automatically by a reception into a receive message buffer or FlexRay internal FIFO or by a change in the slot status. It can be initiated also by a specific user transfer request.

The input and output data transfer can be activated independently. When the input data transfer is activated the application should not directly access message buffers using the FlexRay input buffer. When the output data transfer is activated the application should not directly access message buffers using the FlexRay output buffer.

### 20.3.16.1 Input Data Transfer

When the automatic input data transfer function is enabled, committed input data structures are transferred from the Local RAM/Global RAM to the FlexRay internal message RAM with minimum CPU support.

#### (1) Activation and deactivation

The input data transfer function should be activated before usage. The activation of the input transfer handler initializes the input queue put index (FLXAnFRITS.IPIDX) and get index (FLXAnFRITS.IGIDX) to zero. Also the interrupt status flags in the FLXAnFRITS register (IQEIS and IQFIS) are set to 0.

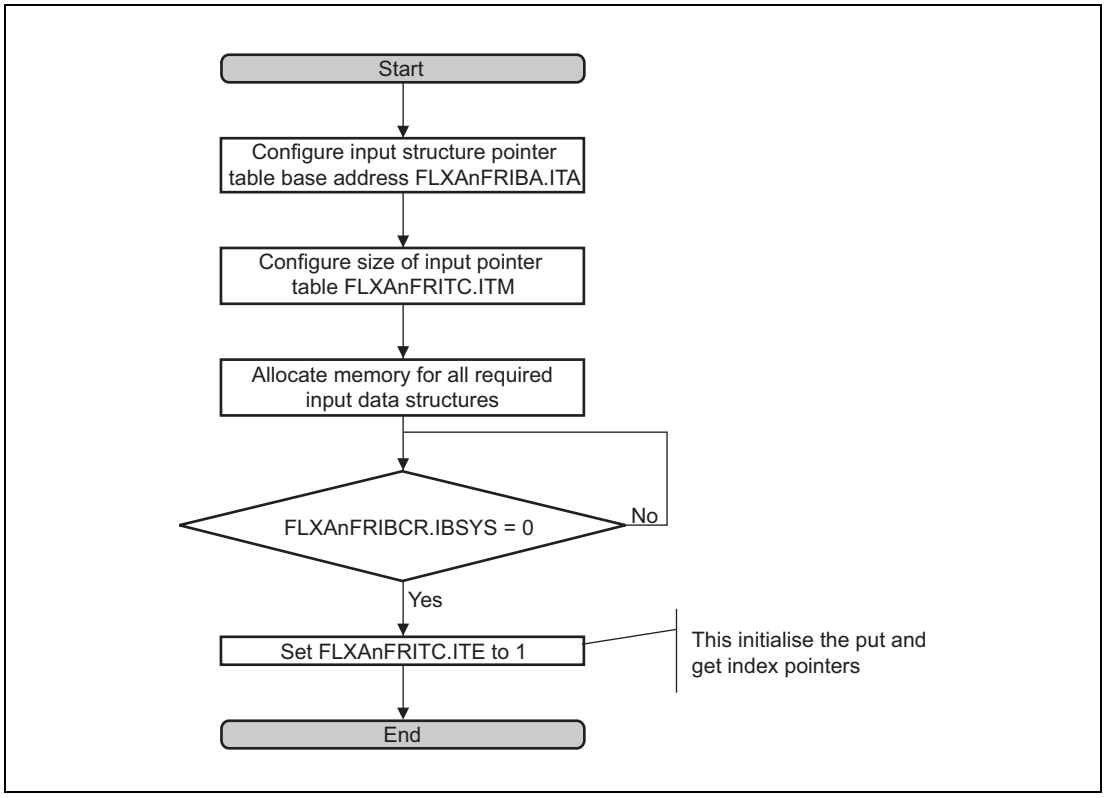



**Figure 20.20 Input transfer enable flow**

A deactivation request of the input transfer function can be made at any time. The input queue put index and the input queue status are maintained independently from the input transfer function state.

Before the transfer function gets disabled (status indicated by FLXAnFRITS.ITS), user requested input transfers and all committed input transfers will be completed.

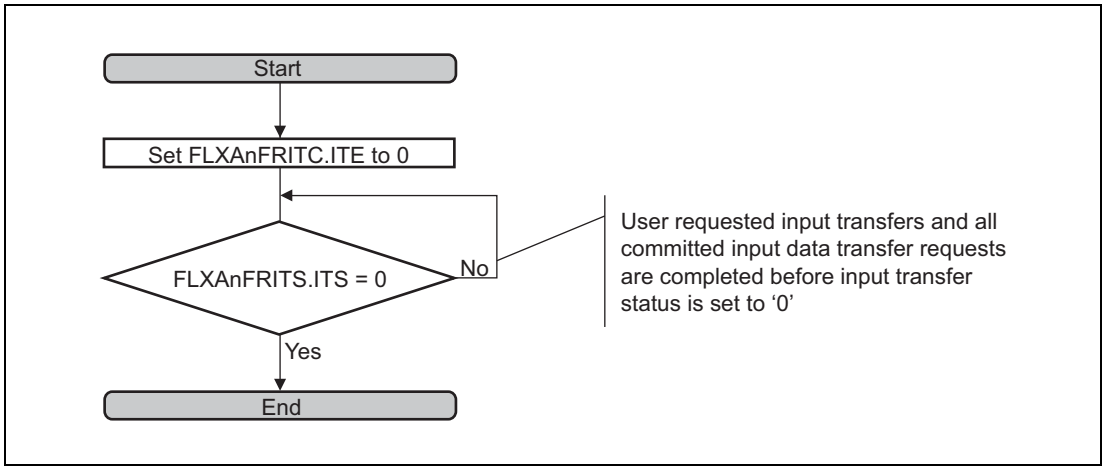



**Figure 20.21 Input transfer disable flow**

### (2) Input data structure

The application has to reserve a location in the Local RAM/Global RAM to provide the content for message buffer configuration (input data structure).

The location of this input data structure needs to be defined by an input data structure pointer also located in the Local RAM/Global RAM.

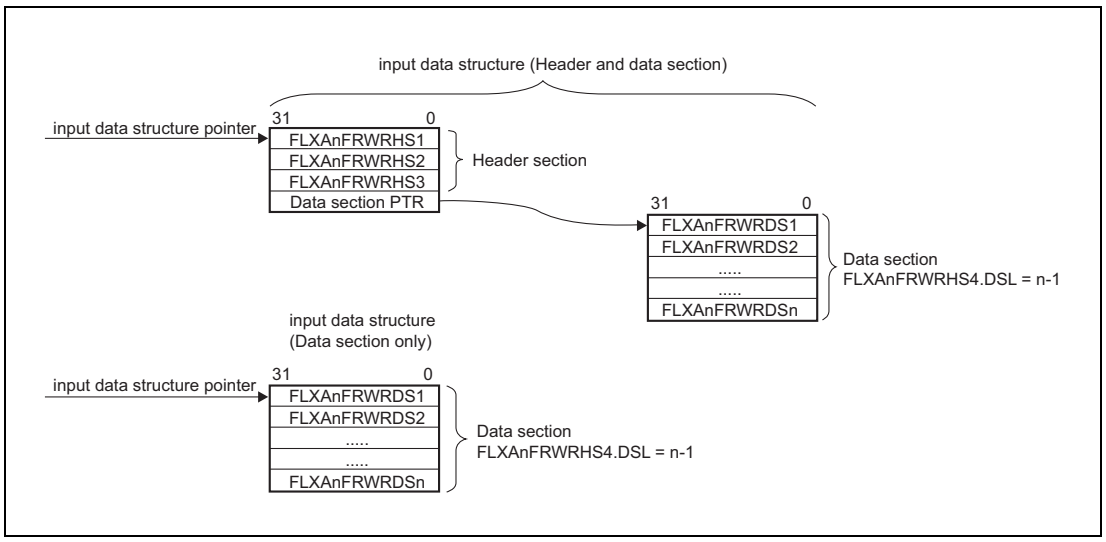



**Figure 20.22 Input data structure**

In general the input data structure consists of two sections, the header and the data section.

The header section consists of FLXAnFRWRHS1, FLXAnFRWRHS2, FLXAnFRWRHS3 and the data section pointer.

For bit alignment and bit function in the header section, see **Section 20.3.13.1, Header Partition**.

Depending on the settings in the control field (FLXAnFRWRHS4) located in the input pointer table, the data structure pointer is a reference to the address of FLXAnFRWRHS1 or FLXAnFRWRDS1. The data structure pointer has to be aligned to a 32 bit address.

If the bit LHS in the address related to FLXAnFRWRHS4 is set to 1 it is required to provide a valid header section. In this case FLXAnFRWRHS1 is the first element of the data structure.

If the bit LHS in the address related to FLXAnFRWRHS4 is set to 0 a header section is not required. In this case FLXAnFRWRDS1 is the first element of the data structure.

If the bit LDS in the address related to FLXAnFRWRHS4 is set to 1 it is required to provide a valid data section. The pointer to the data section is a reference to the address of the first payload long word (FLXAnFRWRDS1) and has to be aligned to a 32 bit address.

If the bit LDS in the address related to FLXAnFRWRHS4 is set to 0 a data section is not required. The data section pointer is not evaluated by the input handler.

The byte order for the FlexRay payload data in the input data structure is determined by FLXAnFROC.BEC. For information about the payload data alignment within the data section see **Section 20.3.13.2, Data Partition** and **Section 20.3.17, Byte Alignment**.

The length of the data section and the size to be allocated in the Local RAM/Global RAM depends on the configuration of the bits DSL in the address related to FLXAnFRWRHS4.

For the transfer into the FlexRay core internal message RAM the number of 16 bit words configured by FLXAnFRWRHS2.PLC is used. The application has to ensure, that a proper number of data words is provided in the Local RAM/Global RAM. In case the buffer is configured by FLXAnFRWRHS2.PLC to hold an odd payload length, the application has to write zero to the last 16 bit of the payload section to ensure that the padding data is all zero.

### (3) Input pointer table

To transfer data from the input data structures located in the Local RAM/Global RAM to the FlexRay internal message RAM the related input data structure pointer and control field needs to be added to the input pointer table which is located in the Local RAM/Global RAM.

The location of the first element of this table is identified by the input pointer table base address (FLXAnFRIBA.ITA). This base address has to be aligned to a 32 bit address.

The maximum number of input requests that can be queued is defined by the Input queue Table Max register (FLXAnFRITC.ITM).

Each Input pointer table entry requires two long words. The required address range of the input pointer table for the queued transfer requests can be calculated by

$$\text{Input pointer table size (byte)} = (((\text{FLXAnFRTTC.ITM} + 1) \times D \times 2) \times 4)$$

**Equation 1**

The input pointer entry for the user requested input transfer should be added after the end of the input pointer table.

The pointer table index related to this entry and hence the number to be written to FLXAnFRUIR.UIDX, is FLXAnFRITC.ITM+1. The address in the input pointer table related to the user requested input transfer (user input address) can be calculated by

User input address = FLXAnFRIBA.ITA + Input pointer table size

**Equation 2**

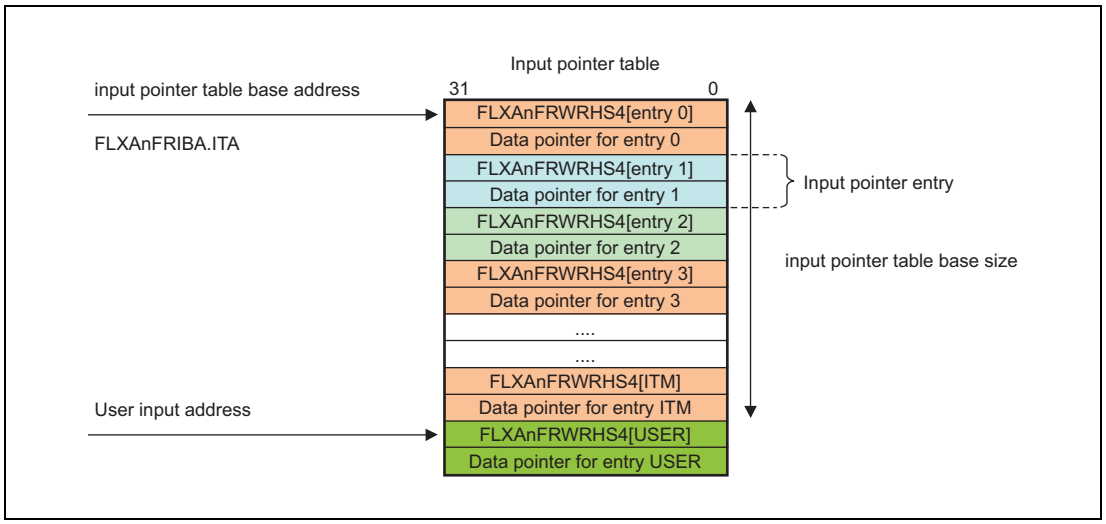



**Figure 20.23 Input pointer table**

The input pointer table holds the control field FLXAnFRWRHS4 and the pointers to the Local RAM/ Global RAM location where the message buffer content (header section and/or data section) is stored.

The application has to write FLXAnFRWRHS4 and the input data structure pointer at the addresses in the input pointer table related to the put index position before a transfer request is initiated.

**FLXAnFRWRHS4:**

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | DSL[5:0] | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | INV | STR | LDS | LHS | — | IMBNR[6:0] | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |

**Table 20.123 FLXAnFRWRHS4 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | When read, an undefined value is read. When writing, always write 0. |
| 21 to 16 | DSL[5:0] | Data Section Length Bit<br>Specifies the length of the data section in terms of 32 bit values. |
| 15 to 12 | Reserved | When read, an undefined value is read. When writing, always write 0. |
| 11 | INV | Invalidate entry Bit<br>0: The data structure is valid and will be transferred to the FlexRay internal message RAM.<br>1: The data structure is invalid. FlexRay internal message RAM is not updated using this input pointer entry. |
| 10 | STR | Set transmission request Bit<br>0: The bit FLXAnFRTXRQi.TXR for the message buffer selected by the bits IMBNR is set to 0. No data from this message buffer is transmitted.<br>1: The bit FLXAnFRTXRQi.TXR for the message buffer selected by the bits IMBNR is set to 1 to release the message buffer for transmission The application should not set the bit STR to 1 for receive buffers. |
| 9 | LDS | Load data section Bit<br>0: No update of data section.<br>1: Data section for the message buffer selected by the bits IMBNR is updated. |
| 8 | LHS | Load header section Bit<br>0: no update of header section.<br>1: Header section for the message buffer selected by the bits IMBNR is updated. |
| 7 | Reserved | When read, an undefined value is read. When writing, always write 0. |
| 6 to 0 | IMBNR[6:0] | Message buffer number to be updated Bit<br>Selects the target message buffer number in the FlexRay internal message RAM for transfer |

Note that the LHS bit should not be set for protected message buffers.

The bit LDS defines if the data section of the message buffer selected by the bits IMBNR should be updated.

If LDS is set to 1 (DSL + 1) 32 bit words of payload data are transferred from the Local RAM/Global RAM to the message buffer selected by the bits IMBNR.

If LDS is set to 0 no payload data is transferred from the Local RAM/Global RAM.

Note that the payload transferred is independent from the configured payload length (bits PLC in the address related to FLXAnFRWRHS2).

The bit INV can be used to invalidate a committed data structure. This bit should be only used to cancel the transfer of committed data structures when the input queue is halted (see **Section 20.3.16.1 (5), Halting the input queue**).

When this bit is set to 1 the message buffer number IMBNR is not updated. When the bit is set to 0 the message buffer number IMBNR is updated.

#### (4) Transfer function of input data structure

To use the input data structure transfer function the input transfer has to be activated (see **Section 20.3.16.1 (1), Activation and deactivation**). The activation process requires the setup of the input pointer table (see **Section 20.3.16.1 (3), Input pointer table**) in order to specify the source location (input data structures) for the data structures to be transferred. When the input transfer gets enabled the get index pointer is initialized to zero.

All FlexRay internal message buffers can be updated using the input transfer queue which is built in the input pointer table. The application has to write the pointer and control field (table entry) to the data structure to be transferred into the input pointer table. For that purpose the application has to maintain a put index for the input pointer table that indicates where the pointer has to be written to.

To commit this table entry to the input handler, the application has to write the target message buffer number to the input queue control register (FLXAnFRIQC.IMBNR). Afterwards the application has to increment the application internal put index.

By writing to the input queue control register the data available flag (FLXAnFRDAi.DA[IMBNR]) is automatically set to 1. The input transfer handler also maintains the put index pointer in the status register (FLXAnFRITS.IPIDX).

In case the input queue gets full (number of queued input transfer requests is equal to the input queue table size) FLXAnFRITS.IQFP and FLXAnFRITS.IQFIS are set to 1. The input queue full condition pending flag (FLXAnFRITS.IQFP) changes from 1 to 0 when there are entries in the input queue available, whereby the input queue full interrupt status flag (FLXAnFRITS.IQFIS) needs to be cleared by the application.

The application should not make any further write access to the bit IMBNR in the FLXAnFRIQC register as long as the bit IQFP in the FLXAnFRITS register is 1.

In case the input queue gets empty (number of queued input transfer requests changes to zero) FLXAnFRITS.IQEIS is set to 1. The input queue empty interrupt status flag (FLXAnFRITS.IQEIS) needs to be cleared by the application.

The transfer of the input data structures to the FlexRay message RAM is controlled by a get index pointer which is handled inside the FlexRay module and flagged in FLXAnFRITS.IGIDX. Note that the index is referring to the input entry and not the address offset in the input pointer table.

If the input queue is not empty, the transfer handler reads out the input pointer table entry of the transfer queue and starts the transfer of the input data structure from the address the input pointer is referring to. When all required data words are transferred to the FlexRay module, the data available flag for the transferred message buffer number is set to 0 and the get index in the transfer handler is incremented by one.

In case of an invalidated data structure (see **Section 20.3.16.1 (5), Halting the input queue**) no FlexRay internal message buffer is updated and the related data available flag is automatically set to 0. The change of the data available flag can be used to confirm the cancellation a transmit request.

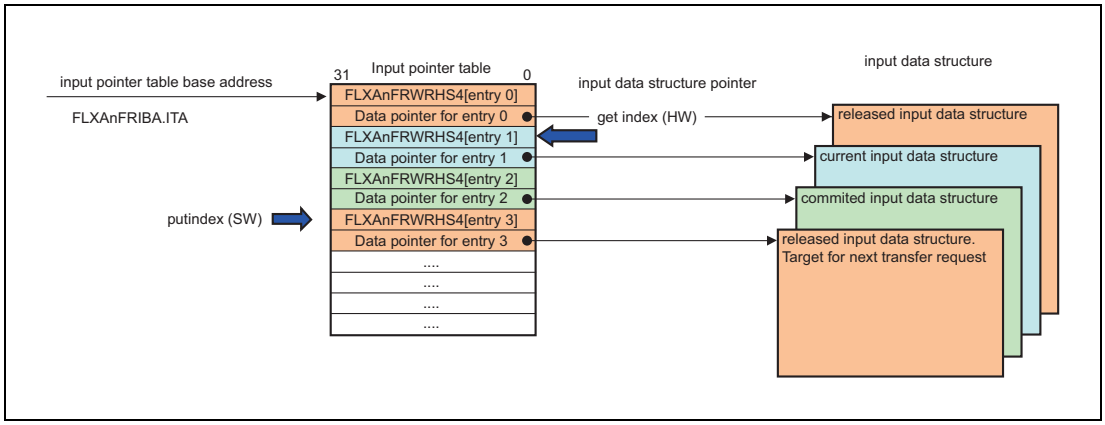



**Figure 20.24 Input Pointer Table**

Receive message buffers can be also configured using the input data transfer by setting up the required header sections and mark only the header section (FLXAnFRWRHS4.LDS = 0, FLXAnFRWRHS4.LHS = 1) to be updated in the FlexRay module.

### (5) Halting the input queue

Committed data structures cannot be removed, but can be invalidated or updated when the input queue is halted.

To cancel data structures already committed to the input queue, the queue can be halted by writing a 1 to FLXAnFRITC.IQHR.

After the ongoing input transfer has been completed the queue is halted and FLXAnFRITS.IQH changes from 0 to 1.

To invalidate an entry of the input queue FLXAnFRWRHS4.INV has to be set to 1. All other bits in FWRHS4 should be unchanged.

Following flow shall be used to analyse whether a committed message has been already transferred to the FlexRay internal message RAM or not.

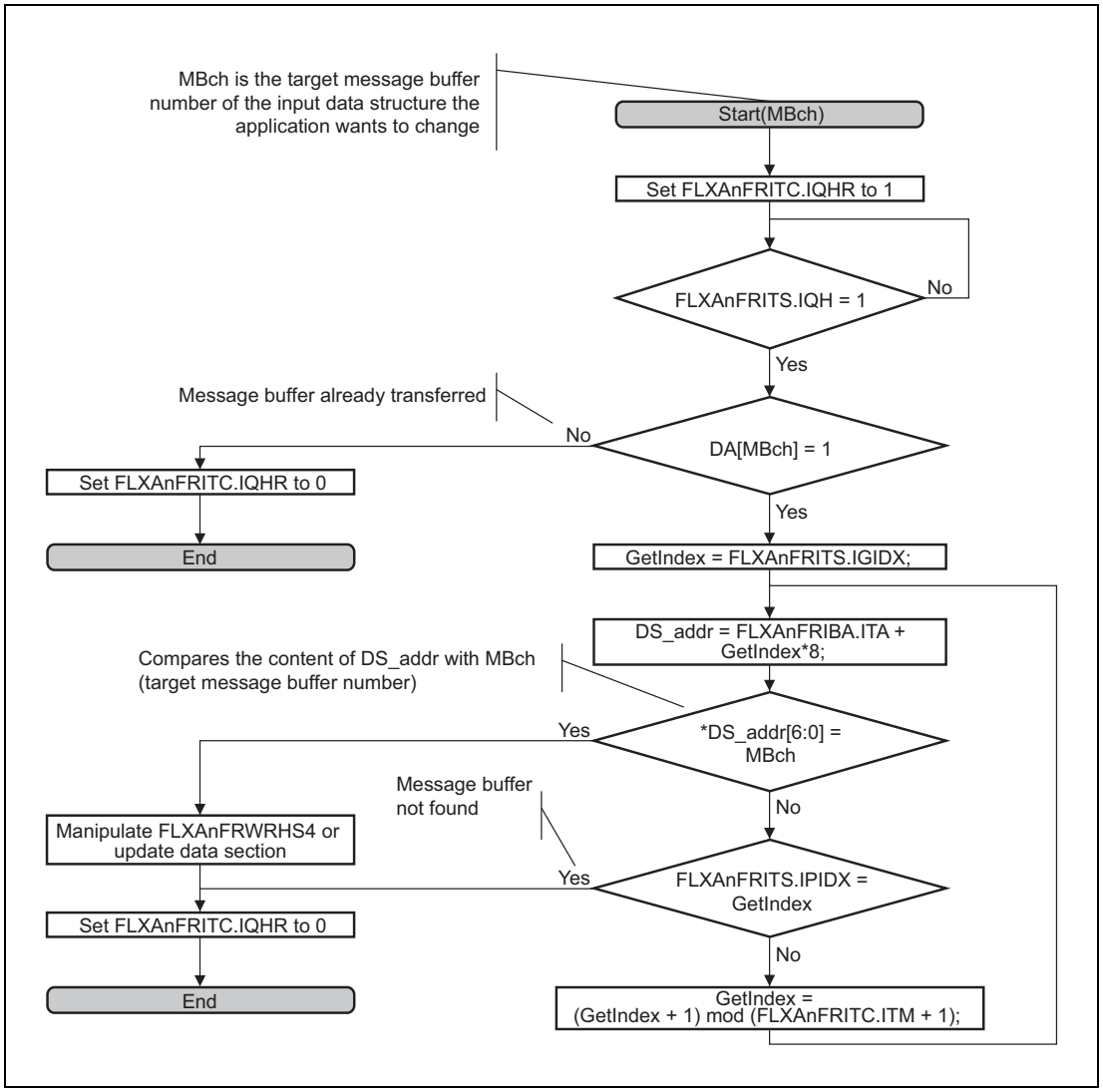



**Figure 20.25 Input Table Analysis**

In case the message buffer was already transferred to the FlexRay internal message RAM, the user input transfer request can be used to bypass the actual queue and update the required message buffer (see **Section 20.3.16.1 (6), Transfer function of user requested input transfers**).

### (6) Transfer function of user requested input transfers

To use this function the input transfer has to be activated (see **Section 20.3.16.1 (1), Activation and deactivation**).

The application is capable, by using FLXAnFRUIR.UIDX, to request a transfer of an input data structure. The user input transfer request is serviced first.

The application has to write the pointer and control field (table entry) to the data structure to be transferred into the input pointer table. The table entry for the user input transfer request should be added after the end of the input pointer table (see **Section 20.3.16.1 (3), Input pointer table**).

To commit this table entry to the input handler, the application has to write the index (FLXAnFRITC.ITM+1) to the user input transfer request register (FLXAnFRUIR.UIDX).

By writing to the user input transfer request register, the user input transfer request pending flag (FLXAnFRITS.UIRP) is automatically set to 1.

As long this flag is 1 the application should not make any further user input transfer requests.

The user input transfer request pending flag (FLXAnFRITS.UIRP) changes from 1 to 0 when the requested input transfer is completed. As next the pending transfers are processed.

### 20.3.16.2 Output Data Transfer

When the output data transfer function is enabled, received messages (either in dedicated message buffers or in the FlexRay receive FIFO) are transferred to the Local RAM/Global RAM by the output data handler. The output data handler can also transfer the message buffer content to the Local RAM/ Global RAM on application request. When enabled the output handler is also capable to initiate a transfer when the message buffer status has changed.

#### (1) Activation and deactivation

The output data transfer function should be activated before usage. The activation of the output transfer hander will initialise the FIFO put and get index pointer and FIFO fill level (FLXAnFROTS.FGIDX and FLXAnFROTS.FFL) to zero, set the bits FLXAnFROTS.FDA, FLXAnFROTS.OWP, FLXAnFROTS.FWP and FLXAnFROTS.UORP to 0. Also the interrupt status flags FLXAnFROTS.OTIS, FLXAnFROTS.FIS, FLXAnFROTS.OWIS and FLXAnFROTS.FWIS) are set to 0.

The activation has no influence to the data available flags (FLXAnFRDAi.DA) which are related to the dedicated buffers; these flags have to be cleared by the application.

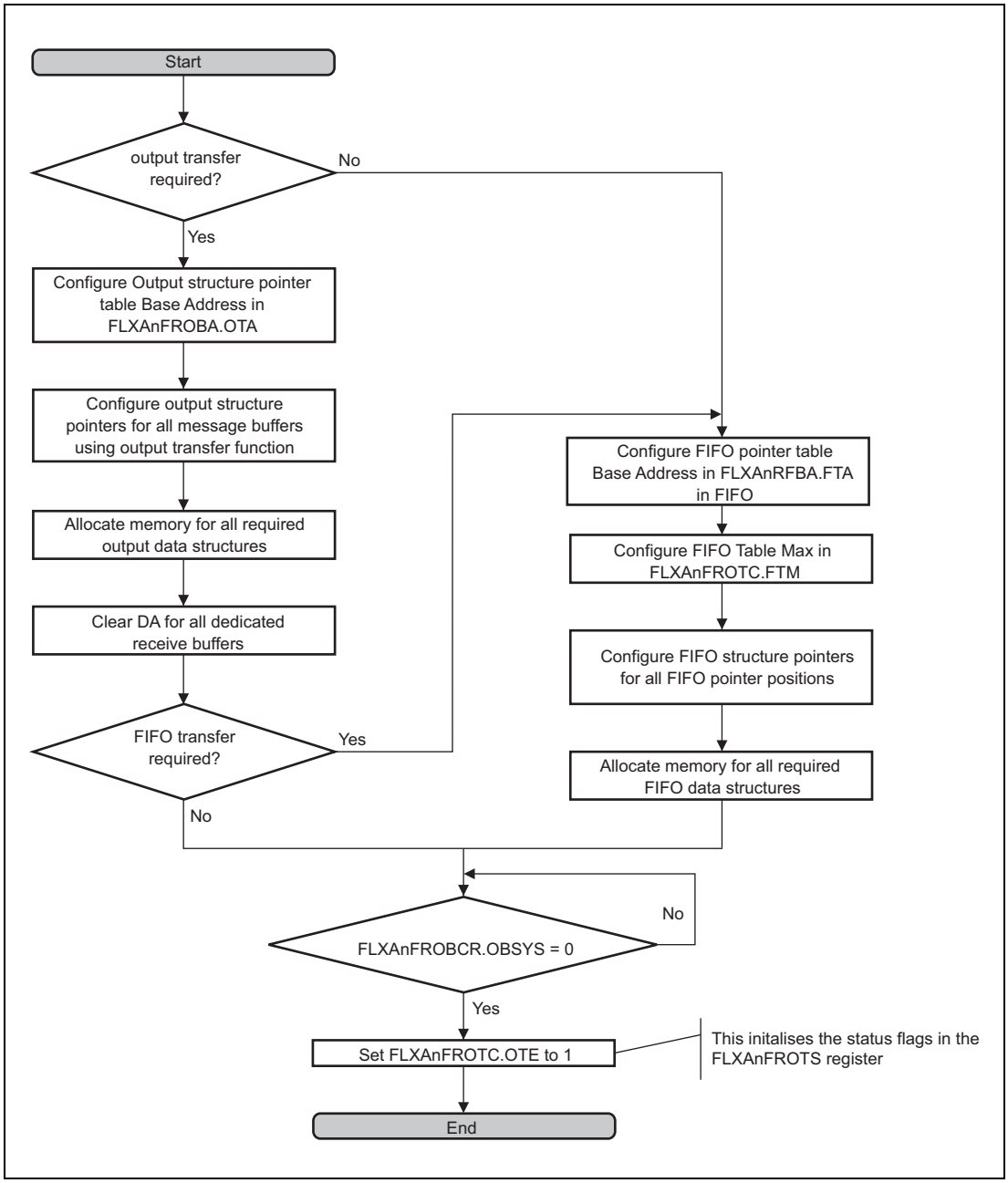



**Figure 20.26 Output Transfer Enable Flow**

A deactivation request of the output data transfer function can be made at any time. An ongoing transfer will be completed and the completion of this transfer will be flagged. During this time FLXAnFROTS.OTS remains 1.

When FLXAnFROTS.OTS changes from 1 to 0, the output transfer function is deactivated. The data available status flags and the FIFO get index are still maintained when the output transfer function is disabled.

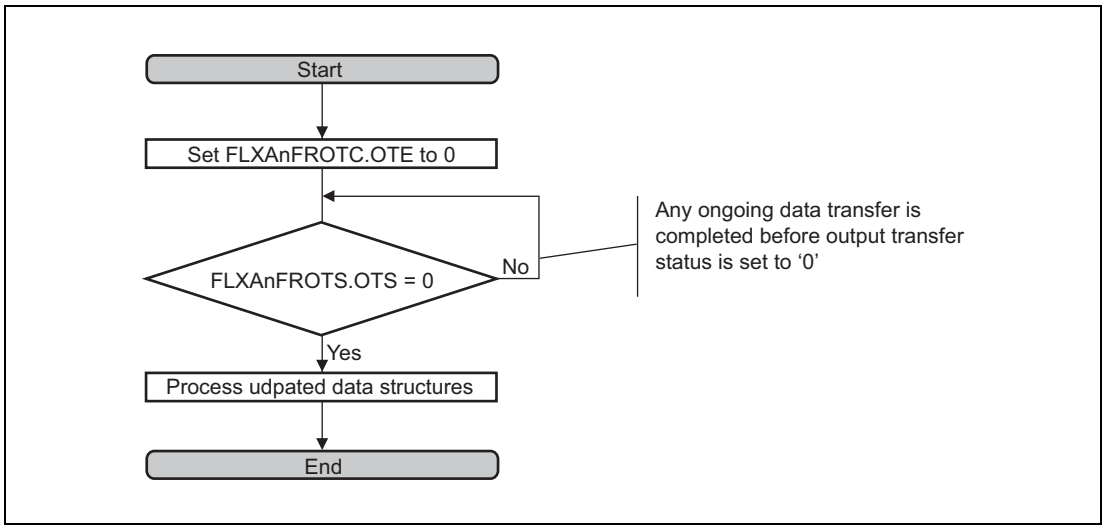



**Figure 20.27 Output Transfer Disable Flow**

### (2) Output transfer data structure

The data in the Local RAM/Global RAM is stored in an output data structure. The location of the output data structures are determined by output data structure pointers also located in the Local RAM/ Global RAM. The output data structure and indexing is visualized in **Figure 20.28**.

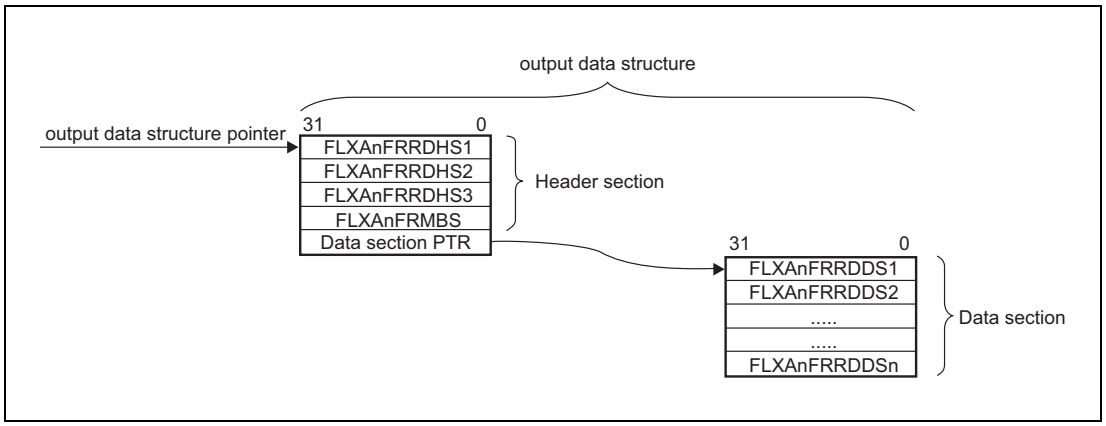



**Figure 20.28 Output Data Structure**

The output data structure consists of two sections, the header and data section. The header section consists of FLXAnFRRDHS1, FLXAnFRRDHS2, FLXAnFRRDHS3, FLXAnFRMBS and the data section pointer. FLXAnFRRDHS1 is the first element of the structure and has to be aligned to a 32 bit address. The data structure pointer is a reference to the address of FLXAnFRRDHS1. For information about the bit alignment and bit function within the header section see **Section 20.3.13.1, Header Partition**.

FLXAnFRRDDS1 is the first element of the data section. The data section pointer is a reference to the address of FLXAnFRRDDS1 and has to be aligned to a 32 bit address. The byte order for the FlexRay payload data in the output data structure is determined by the bit BEC in the FLXAnFROC register. For information about the payload data alignment within the data section see **Section 20.3.13.2, Data Partition**.

The length of the data section as well as the total structure size to be allocated in the Local RAM/ Global RAM depends on the configured payload length (FLXAnFRRDHS2.PLC) of the related message buffer. In case the configured payload length is an odd number of words or the received payload length (FLXAnFRRDHS2.PLR) is smaller than the configured payload length, the remaining data words in the Local RAM/Global RAM are unused and should not be used by the application.

The output data structure is identical for all three kinds of output transfers. In case only the header section is transferred the data section pointer is not evaluated by the output handler and the data section remains unchanged.

### (3) Output pointer table

For the output data transfer function the application needs to setup an output pointer table in the Local RAM/Global RAM. The location of the first element of this table should be programmed into the output pointer table base address (FLXAnFROBA.OTA). This base address has to be aligned to a 32 bit address.

The size of the output pointer table is defined by the maximum of: the last configured dedicated message buffer and the highest message buffer number which will be used for the user output transfer request.

The output pointer table holds pointers (output data structure pointers) to the Local RAM/Global RAM location where a memory space is reserved for the target message buffer content (header section and data section).

There is a fixed linear relationship between the address of the entries in the output pointer table and the number of the related message buffers (see **Figure 20.29**): the output pointer table starts with the entry for message buffer number 0 at the address configured in FLXAnFROBA.OTA and continues in ascending order for each following message buffer number, by 32 bit aligned address (e.g. message buffer 1 at address OTA + 4, message buffer 2 at address OTA + 8, etc.) for all possible message buffers.

When a set ND bit is the only transfer condition (FLXAnFROTC.OTCS is set to 0) only message buffers configured as a dedicated receive buffer or that will be used for user output transfer requests need have valid pointer entries.

When a set ND bit or a set MBSC bit is the transfer condition (FLXAnFROTC.OTCS is set to 1) all dedicated receive buffer and dedicated transmit buffers need to have valid pointer entries.

### (4) FIFO output pointer table

The FlexRay module internal FIFO can be extended by a queued buffer structure in the Local RAM/ Global RAM.

If the FlexRay module internal FIFO is used the application needs to setup the FIFO output pointer table. The location of the first element of this table is identified by the FIFO pointer table base address (FLXAnFRFBA.FTA). This base address has to be aligned to a 32 bit address.

The size of the FIFO pointer table and hence the maximum number of messages that can be added to the queue, is defined by FIFO Table Max (FLXAnFROTC.FTM).

The FIFO pointer table holds pointers (output data structure pointers) to the Local RAM/Global RAM location where a memory space is reserved the target message buffer content (header section and data section). For each table entry a data pointer shall be configured in this table.

### (5) Transfer function of dedicated message buffers

To use this transfer function the output transfer has to be activated (see **Section 20.3.16.1 (1), Activation and deactivation**). The activation process requires to setup the output pointer table (see **Section 20.3.16.2 (3), Output pointer table**) in order to specify the destination location (output data structures) for the data to transfer. **Figure 20.29** shows how the output pointer table references the output data structures.

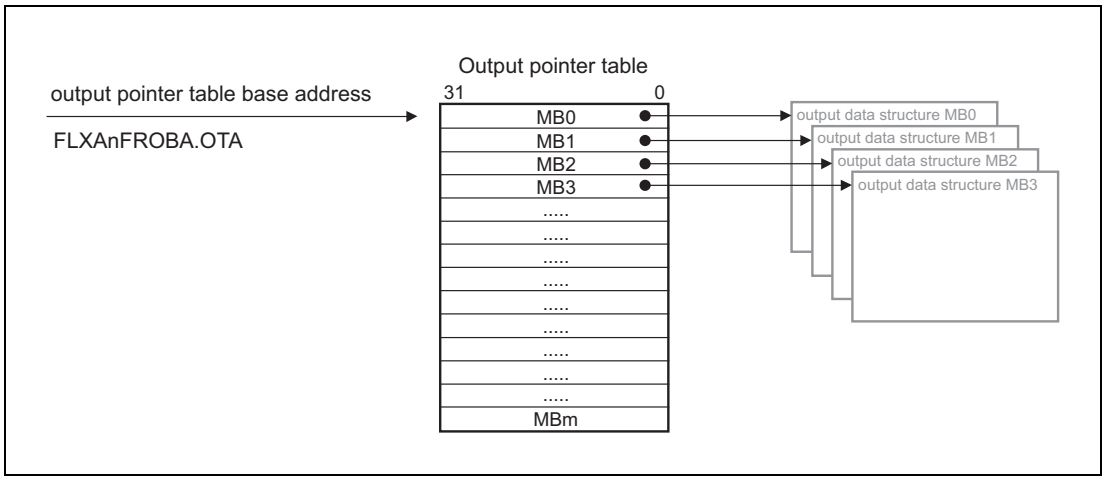



**Figure 20.29 Output Data Structure and Indexing**

With FLXAnFROTC.OTCS the output transfer condition can be selected between the ‘New data only mode’ and the ‘New data and status changed mode’.

In the ‘New data only mode’ an output data transfer is initiated when a valid FlexRay data frame has been stored into a dedicated receive buffer which causes the related ND flag in the FLXAnFRNDAT register to set. The ND flag in the FLXAnFRNDAT register is automatically set to 0 during the transfer procedure. The header section is also transferred and hence the MBSC flag in the FLXAnFRMBSC register is set to 0.

In the ‘New data and status changed mode’ an output data transfer is initiated as described in the ‘New data only mode’. In addition an output data transfer is initiated when only the message buffer status has been changed which causes the related MBSC flag in the FLXAnFRMBSC register to be set. In this case only the header section is transferred. The MBSC flag in the FLXAnFRMBSC register is automatically set to 0 during the transfer procedure.

After transferring the message buffer data from the FlexRay internal message RAM to the output data structure the corresponding data available flag in the FLXAnFRDAi (i = 0 to 3) registers is set to 1. The update of the output data structure is also flagged by the setting of the output transfer interrupt status flag (FLXAnFROTS.OTIS).

As long as the data available flag remains 1 the corresponding output data structure will not be updated.

In the case

- the data available flag is 1 and a valid received message was stored or
- when FLXAnFROTC.OTCS is 1 and the message buffer status was updated,

the output transfer warning interrupt flag (FLXAnFROTS.OWIS) is set to 1 notifying the application that new data is available but the output data structure transfer cannot be processed. In addition FLXAnFROTS.OWP is set to 1 that continuously flags that status of the output transfer warning condition.

If a valid receive message in the FlexRay internal message RAM is overwritten by an additional receive message, the message lost flag (FLXAnFRMBS.MLST) is set to 1. This flag can be evaluated after the message buffer has been transferred into an output data structure.

Following sections are giving a guidance how output data structures can be handled.

(a) Data section copy method

One option is to copy the information from the output data structure to a different location of the Local RAM/Global RAM and then release the output data structure by clearing the related data available flag. The application should use the copied information for further processing.

(b) Data structure pointer method

A different option is to modify the output data structure pointer in the output pointer table and to release the output data structure by clearing the related data available flag. The changed output data pointer should refer to a free data structure. The application should use the old data structure for further processing.

(c) Data section pointer method

A third option is to modify the data section pointer in the output data structure and to release the output data structure by clearing the related data available flag. The changed data section pointer should refer to a free memory area. The application should use the old data section for further processing by forwarding the data section pointer.

### (6) Transfer function of FIFO message buffers

To use this buffer transfer function the output transfer has to be activated (see **Section 20.3.16.1 (1), Activation and deactivation**). The activation process requires the setup of the FIFO pointer table (see **Section 20.3.16.2 (4), FIFO output pointer table**) in order to specify a location in the Local RAM/Global RAM reserved for the storage of the required output data structures.

A FIFO data transfer is initiated when a valid FlexRay data frame has been stored in the FlexRay internal FIFO.

After transfers from the internal FIFO to the output data structure, the FIFO interrupt status flag (FLXAnFROTS.FIS) and FLXAnFROTS.FDA are set to 1. The bit FLXAnFROTS.FIS can be used as an interrupt source. The bit FLXAnFROTS.FDA indicates that the FIFO is not empty.

Up to FLXAnFROTS.FTM output structures configured in the FLXAnFROTC register can be queued.

The transfer to the extended FIFO buffer structure is controlled by index pointers. This put index is controlled by the FIFO transfer handler and is incremented after transferring a message to the output data structure.

The FIFO reception handler also maintains a get index which is flagged in FLXAnFROTS.FGIDX. The value of this get index is known by the application by either reading the status or maintaining a software variable. The get index (the value after a reset is zero) is incremented by one when the application releases the oldest entry of the FIFO queue by writing 1 to FLXAnFROTS.FDA. By comparing the put index and the get index the FIFO handler knows about the current fill level of the queued buffer structure.

The current FIFO fill level is flagged in FLXAnFROTS.FFL. When FLXAnFROTS.FDA is 1, there is at least one entry in the FIFO queue.

In case the queued buffer structure in the Local RAM/Global RAM is full (FLXAnFROTS.FFL = FLXAnFROTC.FTM+1), no further transfers are initialised, new messages remain in the FlexRay internal FIFO and the FIFO transfer warning interrupt status flag (FLXAnFROTS.FWIS) is set to 1.

In case the FlexRay internal FIFO structure becomes full, messages in the FlexRay internal FIFO structure may get overwritten. The related status flags and configuration registers of the FlexRay core module can be used to generate desired warning notifications.

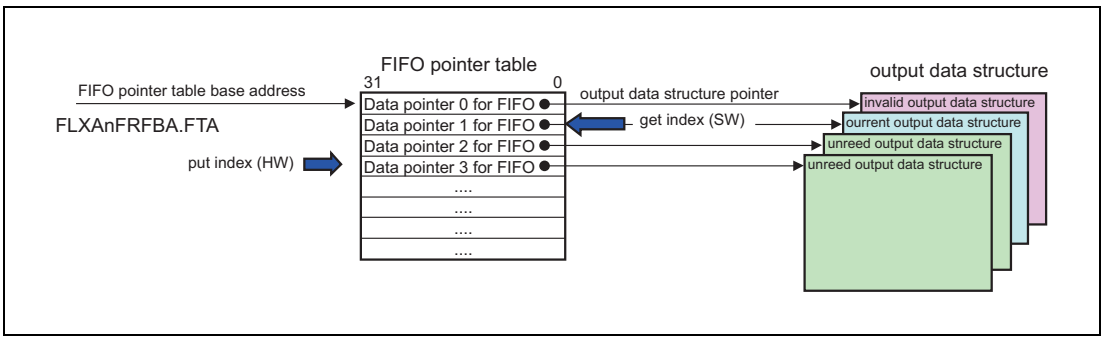



**Figure 20.30 FIFO Pointer Table**

### (7) Transfer function of user output transfer requests

To use this transfer function the output transfer has to be activated (see **Section 20.3.16.1 (1), Activation and deactivation**). The activation process requires to setup the output pointer table (see **Section 20.3.16.2 (3), Output pointer table**) in order to specify the location in the Local RAM/ Global RAM reserved for the transfer of the data (output data structures).

The application is capable, by using FLXAnFRUOR.UMBNR, to request a transfer of dedicated message buffer to an output data structure. Except in CONFIG state, message buffers which are part of the FlexRay internal FIFO should not be requested.

The header section is always transferred to the output data structure. The transfer of the data section can be enabled by setting FLXAnFRUOR.URDS to 1. The selected message buffer content is stored in the output data structure location determined by the pointers in the output pointer table.

The data available status and transfer blocking by the DA bits in the FLXAnFRDAi register is also used for the user requested transfers. Therefore the DA bit in the FLXAnFRDAi register related to the requested buffer number (FLXAnFRUOR.UMBNR) should be released before making the transfer request.

After writing to FLXAnFRUOR.UMBNR, the bit FLXAnFROTS.UORP is set to 1 to indicate that there is a pending user transfer request. When the transfer has been processed the bit FLXAnFROTS.UORP is set to 0, the bit FLXAnFROTS.OTIS is set to 1 and the DA bit in the FLXAnFRDAi register related to the requested buffer number (FLXAnFRUOR.UMBNR) is set to 1.

User output transfer requests cannot be queued. The application should check the bit FLXAnFROTS.UORP before writing to FLXAnFRUOR.UMBNR.

User output transfer requests should not be made for message buffers which are pending in the input transfer queue.

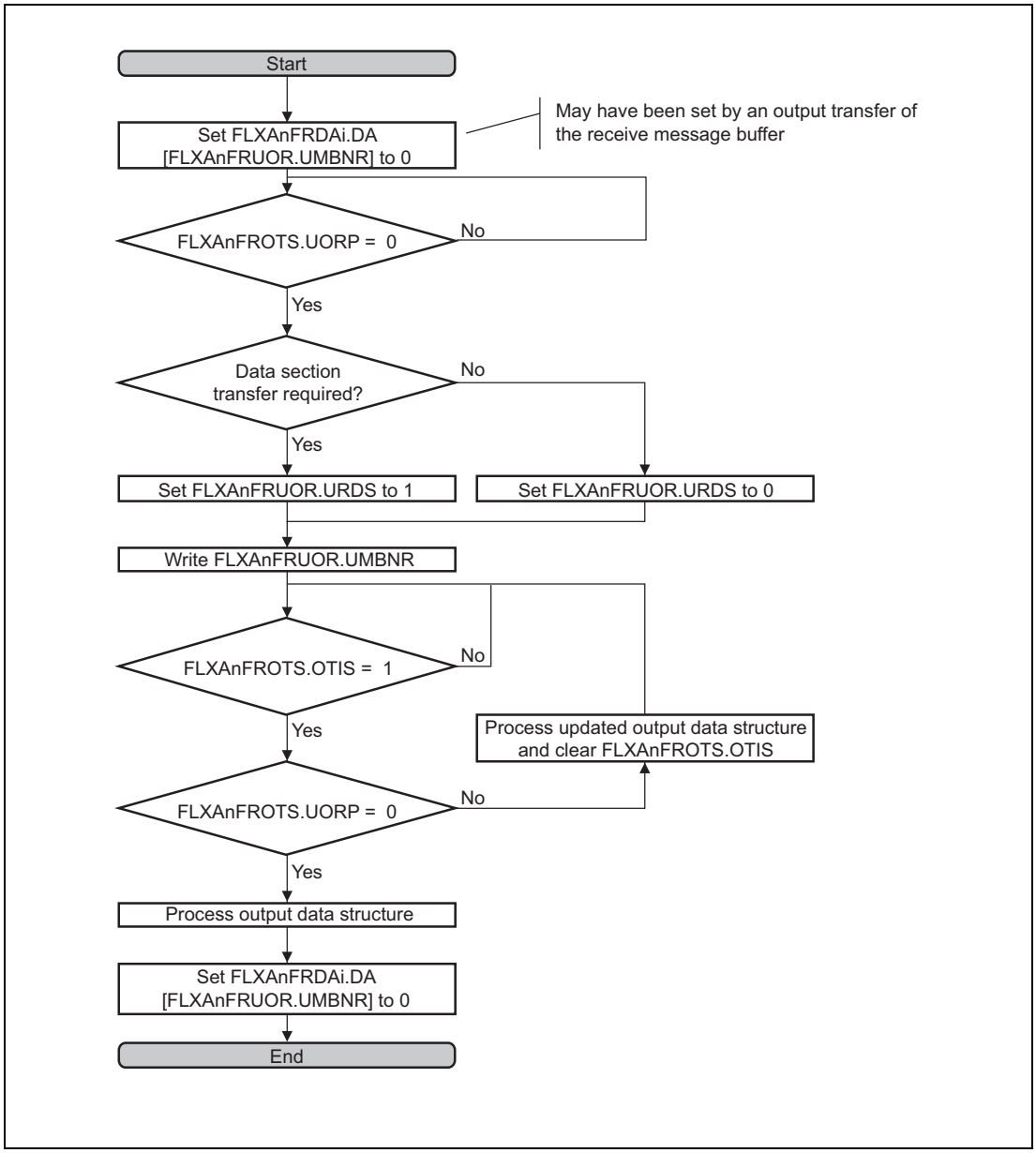



**Figure 20.31 User Output Transfer Request Flow**

Note that it may be possible that the data structure addressed by a user request is being updated due to a receive message buffer update (which causes the bit DA in the FLXAnFRDAi register being set). This set DA flags inhibits the user output transfer request. Therefore polling FLXAnFROTS.UORP is not a secure method to identify when the transfer of a requested message buffer has been completed. The bits FLXAnFROTS.OTIS or the DA bit in the FLXAnFRDAi register can be used instead. The exact flow depends on the software architecture.

### 20.3.16.3 Data Structure Transfer Scheduling

Cyclically the different types of transfer requests are checked. In order to guarantee a certain transfer time the different types of transfers have different priorities.

Use requested input transfers have highest priority followed by the transfer of data structures committed into the active input transfer queue. No new output transfer will be started as long as there is a pending input transfer request.

The three output transfer request types are checked in a specific order:

#### (1) All dedicated message buffers in ascending order

When FLXAnFROTC.OTCS is set to 0, set flags in the FLXAnFRNDATi register are causing a transfer of the message buffer to the output data structure if the destination area is free (DA bit in the FLXAnFRDAi register is 0).

When FLXAnFROTC.OTCS is set to 1, set flags in the FLXAnFRNDATi register or set flags in the FLXAnFRMBSCn register are causing a transfer of the message buffer to the output data structure if the destination area is free (DA bit in the FLXAnFRDAi register is 0).

#### (2) FlexRay internal FIFO

When the FlexRay internal FIFO is not empty and there is a free destination area, one FIFO message is transferred into the output data structure addressed by the FIFO pointer table.

#### (3) User output request

If there is a pending user output transfer request, one message buffer is transferred into the corresponding output data structure.

The check sequence is suspended when an input transfer occurs.

### 20.3.16.4 Behavior in Case of Data Transfer Access Error

The memory areas accessed by the data transfer function may be protected by a memory protection unit (MPU). When the MPU flags an access to a protected address caused by an input or output transfer, an access error event is generated and the related bit in the FLXAnFRAES register is set.

The ongoing transfer is immediately terminated but succeeding transfers are processed and may generate further access errors. Any following access errors are only flagged in FLXAnFRAES.MAE. The other status flags are not updated.

#### (1) Access error during input transfer

When an access error occurs during an input transfer:

- The ongoing transfer is immediately terminated. The FlexRay internal message RAM will not be updated
- The address, the FlexRay module wanted to access to, is captured in the FLXAnFRAEA register
- FLXAnFRAES.IAE is set to 1
- The input pointer table index is flagged in FLXAnFRAES.EIDX
- In case of an normal input transfer the to the transfer related DA flag in the FLXAnFRDAi register is set to 0
- In case of an user input transfer request FLXAnFRITS.UIRP is set to 0

With the given status information the application is able to identify and correct the faulty data structure.

In addition the application needs to clear the input access error flag (FLXAnFRAES.IAE).

### (2) Access error during output transfer

When an access error occurs during an output transfer:

- The ongoing transfer is immediately terminated but the update of the data structure may have started.
- The address, the FlexRay module wanted to access to, is captured in the FLXAnFRAEA register
- FLXAnFRAES.OAE is set to 1
- The output pointer table index is flagged in FLXAnFRAES.EIDX
- In case of an normal output transfer the to the transfer related DA flag in the FLXAnFRDAi register remains 0 and no output transfer interrupt is generated
- In case of an user output transfer request FLXAnFROTS.UORP is set to 0

With the given status information the application is able to identify and correct the faulty data structure. The data structure in the Local RAM/Global RAM cannot be treated as valid.

In addition the application needs to clear the output access error flag (FLXAnFRAES.OAE).

The FlexRay module internal transfer of the message buffer is completed before the Local RAM/ Global RAM access error is detected. The output transfer will not be re-initiated. To avoid loss of data, the application can perform a user output transfer request of this message buffer to a correct Local RAM/Global RAM location.

### (3) Access error during FIFO transfer

When an access error occurs during an FIFO transfer:

- The ongoing transfer is immediately terminated
- The address, the FlexRay module wanted to access to, is captured in the FLXAnFRAEA register
- FLXAnFRAES.FAE is set to 1
- The FIFO pointer table index is flagged in FLXAnFRAES.EIDX
- The FIFO index pointer are not changed and hence the FIFO status flags are unchanged

With the given status information the application is able to identify and correct the faulty data structure.

In addition the application needs to clear the FIFO access error flag (FLXAnFRAES.FAE).

The data in the Local RAM/Global RAM cannot be treated as valid and is not released to the application. The message cannot be recovered.

### 20.3.16.5 Behaviors in Case of RAM Read Errors

The FlexRay internal message RAM has an ECC checking mechanism. In case an uncorrectable RAM read error occurs, the application has to analyze the status in the FLXAnFRMHDS register and react as described in **Section 20.3.16.3, Data Structure Transfer Scheduling**. The input and output transfer handler reacts also on these errors detected in the message RAM when the error is related to an active transfer.

In addition, the Temporary Buffer RAM A and Temporary Buffer RAM B have an ECC checking mechanism as well. An uncorrectable RAM read errors does not impact the data transfer functionality but have to be handled as described in **Section 20.3.13.1 (4), Message Buffer Status FLXAnFRMBS (word 3)**.

In all cases, data causing a read error is never transferred to the Local RAM/Global RAM. If there is no recovery available in the application, the message is lost.

#### (1) Read error during transfer from TBF to MBF

This internal transfer is done for each valid FlexRay message received.

A read error can only occur when reading the header section in the FlexRay Message RAM (see read error flags in FLXAnFRMHDS). In this case, the message buffer needs to be re-configured.

For dedicated receive message buffers, the related ND flag in the FLXAnFRNDAT register will not get set. Consequently the affected message buffer will not be transferred to the output data structure.

For the FlexRay internal FIFO buffers, the ND flag in the FLXAnFRNDAT register is not set but the FlexRay internal FIFO put index is incremented. Due to this, a transfer procedure from the FlexRay internal FIFO buffer to the output buffer is started. However, if the read error is still present in the header section, updating of the output data structure will not start (see **Section 20.3.16.5 (2), Read error during transfer from MBF to OBF**); thus the data in the Local RAM/Global RAM remains correct.

Note that the correction or any other reconfiguration of FIFO related to message buffers while there are pending FIFO transfers may result in incorrect data in the Local RAM/Global RAM. It is strongly recommended to deactivate the output data transfer before starting the reconfiguration and flush the FlexRay internal FIFO before reactivation of the output data transfer.

#### (2) Read error during transfer from MBF to OBF

This internal transfer is done for every output data transfer (dedicated reception, FIFO, user requested).

A read error can occur in the header and data section (see read error flags in FLXAnFRMHDS). In both cases the message gets lost. If the error is located in the header section, the message buffer needs to be re-configured. If the error is located in the data section, the error is corrected with the next data section update.

When a read error occurs during the transfer from the message RAM to the output buffer, the output data structure will not be updated and the data available will not be set to 1. The FIFO put index and the FIFO fill level are not changed also.

In case of user output transfer request, FLXAnFROTS.UORP is set to 0 even if there was no update of the output data structure.

#### (3) Read error during transfer from IBF to MBF

This internal transfer is done for every input data transfer.

A read error can occur only when there is no update of the header section requested (the bit LHS in FLXAnFRWRHS4 is set to 0) due to the reading of the header section from the message RAM (see read error flags in FLXAnFRMHDS). In this case, the message buffer needs to be re-configured.

When a read error occurs during the input data transfer, the actually transferred message in the input queue gets lost.

#### (4) Message RAM read errors

Read errors when reading the header section are flagged in the FLXAnFRMHDS register.

Depending on the buffer type and set buffer protection, a reconfiguration of the message buffer may not be possible.

The input transfer function cannot be used to reconfigure a locked message buffer using the method described in **Section 20.3.13.4 (3), Temporary Unlocking of Header Section**.

Before reconfiguring a locked buffer, the user should disable the input transfer function and the output transfer function.

## 20.3.17 Byte Alignment

The alignment of the bytes received by the FlexRay protocol and the alignment of the bytes required by the application may be different. The FlexRay module provides with FLXAnFROC.BEC a byte alignment function to support different byte ordering styles.

**Figure 20.32** shows the payload byte alignment in a FlexRay frame.

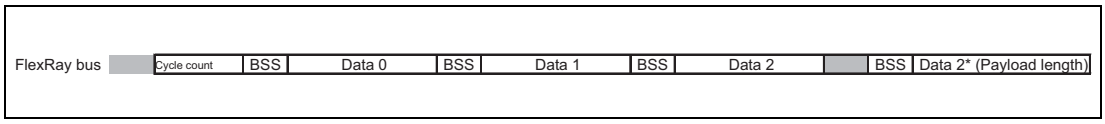



**Figure 20.32 Byte Alignment on the FlexRay Bus**

### 20.3.17.1 Little Endian Alignment

When FLXAnFROC.BEC is 0, the byte alignment is set to Little Endian.

#### (1) FLXAnFRNMVm (m = 1 to 3)

The byte alignment of the NMV bytes is as below.

| Bit | 31 30 29 28 27 26 25 24 | 23 22 21 20 19 18 17 16 | 15 14 13 12 11 10 9 8 | 7 6 5 4 3 2 1 0 |
|---|---|---|---|---|
| FLXAnFRNMV1 | Data 3 | Data 2 | Data 1 | Data 0 |
| FLXAnFRNMV2 | Data 7 | Data 6 | Data 5 | Data 4 |
| FLXAnFRNMV3 | Data 11 | Data 10 | Data 9 | Data 8 |

#### (2) FLXAnFRWRDSx (x = 1 to 64)

The byte alignment for the message payload in the FlexRay input buffer and the input data structure is as below.

FLXAnFRWRDSxMD[7:0] = $Data_{4x-4}$

FLXAnFRWRDSxMD[15:8] = $Data_{4x-3}$

FLXAnFRWRDSxMD[23:16] =$Data_{4x-2}$

FLXAnFRWRDSxMD[31:24] = $Data_{4x-1}$

Transmission order on the FlexRay bus is FLXAnFRWRDSx.MD[7:0], FLXAnFRWRDSx.MD [15:8], FLXAnFRWRDSx.MD [23:16], FLXAnFRWRDSx.MD [31:24] with the most significant bit (MSB) transmitted first.

#### (3) FLXAnFRRDDSx (x = 1 to 64)

The byte alignment for the message payload in the FlexRay output buffer and the output data structure is as below.

FLXAnFRRDDSxMD[7:0] = $Data_{4x-4}$

FLXAnFRRDDSxMD[15:8] = $Data_{4x-3}$

FLXAnFRRDDSxMD[23:16] =$Data_{4x-2}$

FLXAnFRRDDSxMD[31:24] = $Data_{4x-1}$

Reception order on the FlexRay bus is FLXAnFRRDDSx.MD[7:0], FLXAnFRRDDSx.MD [15:8], FLXAnFRRDDSx.MD [23:16], FLXAnFRRDDSx.MD [31:24] with the most significant bit (MSB) received first.

### 20.3.17.2 Big Endian Alignment

When FLXAnFROC.BEC is 1, the byte alignment is set to Big Endian.

#### (1) FLXAnFRNMVm (m = 1 to 3)

The byte alignment of the NMV bytes is as below.

| Bit | 31 30 29 28 27 26 25 24 | 23 22 21 20 19 18 17 16 | 15 14 13 12 11 10 9 8 | 7 6 5 4 3 2 1 0 |
|---|---|---|---|---|
| FLXAnFRNMV1 | Data 0 | Data 1 | Data 2 | Data 3 |
| FLXAnFRNMV2 | Data 4 | Data 5 | Data 6 | Data 7 |
| FLXAnFRNMV3 | Data 8 | Data 9 | Data 10 | Data 11 |

#### (2) FLXAnFRWRDSx (x = 1 to 64)

The byte alignment for the message payload in the FlexRay input buffer and the input data structure is as below.

FLXAnFRWRDSxMD[7:0] = $Data_{4x-1}$

FLXAnFRWRDSxMD[15:8] = $Data_{4x-2}$

FLXAnFRWRDSxMD[23:16] = $Data_{4x-3}$

FLXAnFRWRDSxMD[31:24] = $Data_{4x-4}$

Transmission order on the FlexRay bus is FLXAnFRWRDSx.MD[7:0], FLXAnFRWRDSx.MD [15:8], FLXAnFRWRDSx.MD [23:16], FLXAnFRWRDSx.MD [31:24] with the most significant bit (MSB) transmitted first.

#### (3) FLXAnFRRDDSx (x = 1 to 64)

The byte alignment for the message payload in the FlexRay output buffer and the output data structure is as below.

FLXAnFRRDDSxMD[7:0] = $Data_{4x-1}$

FLXAnFRRDDSxMD[15:8] = $Data_{4x-2}$

FLXAnFRRDDSxMD[23:16] = $Data_{4x-3}$

FLXAnFRRDDSxMD[31:24] = $Data_{4x-4}$

Reception order on the FlexRay bus is FLXAnFRRDDSx.MD[7:0], FLXAnFRRDDSx.MD [15:8], FLXAnFRRDDSx.MD [23:16], FLXAnFRRDDSx.MD [31:24] with the most significant bit (MSB) received first.

### 20.3.18 FlexRay NTU supervision

In case of normal operating condition the FlexRay is sending NTU (Network Timing Unit) HW triggers to the System timer, which enables SW to synchronize the drivers job list processing with very low effort.

For keeping SW driver synchronization (e.g. Autosar) while FlexRay communication is not working properly or is temporarily not available (e.g. loss of NTU trigger), the system timer can support SW synchronization by emulating the FlexRay NTU trigger with a configurable frequency.

To support NTU supervision the E-Ray IP is providing a dedicated output signal, which is connected to the NTU Clock Selector. This selector provides an input signal for STM0 (in case there are two STM timers, only the first one is connected to NTU Tick). The frequency of the NTU clock signal and of the Backup clock, derived from the low jitter PLL used for FlexRay communication, are continuously compared by the NTU loss monitor.

The NTU clock of the FlexRay IP is equal to a macrotick (MT) or cycle tick (CT). The backup clock can be configured to the same frequency. Additionally the eray_sclk and the PLL clock used for NTU loss monitor are the same signal. The frequency of the backup clock has to be configured during initialisation phase and cannot be changed during normal operation.

#### 20.3.18.1 Window function

The user can configure a “NTU loss” window. If the monitored NTU tick is not detected during the open window, an error is generated:

The NTU loss monitor operation:

- NPRS, NBPRS and NCPRS divider blocks for each input clock. Dividing factor is based on the setting in the respective registers
- 12bit compare unit
- The counter inside compare unit counts number of NTU signal’s rising edges during 16 clock periods of base clock
- If the counted NTU ticks value is outside of the threshold values set in the respective registers (upper and lower) an INTC signal occurs.
- After that user can change the selected NTU source

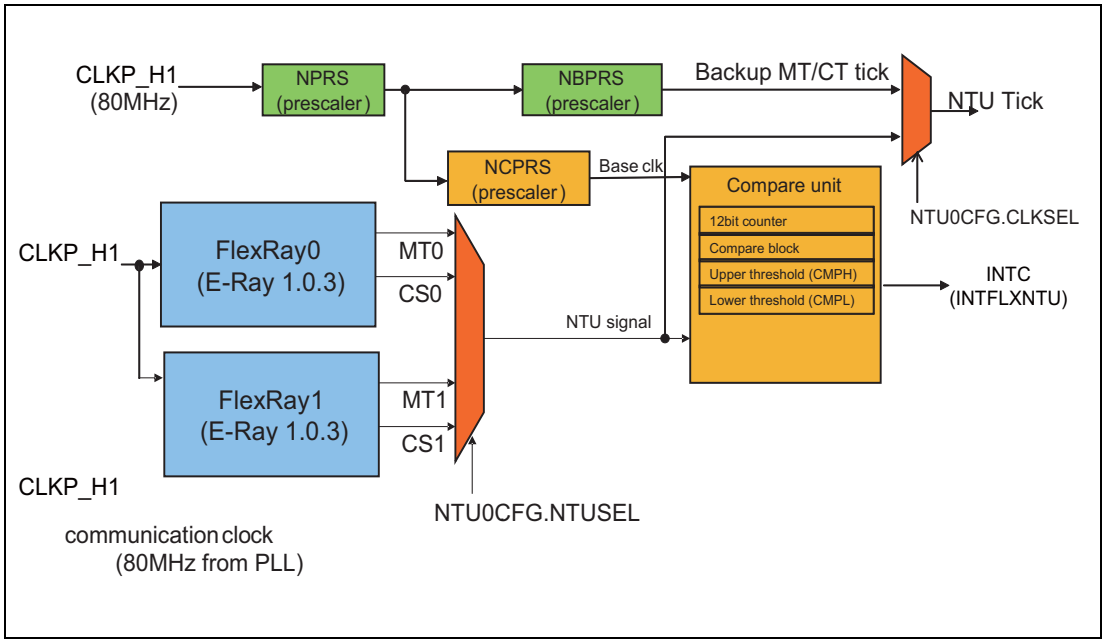



**Figure 20.33 Block diagram of NTU generation for STM**

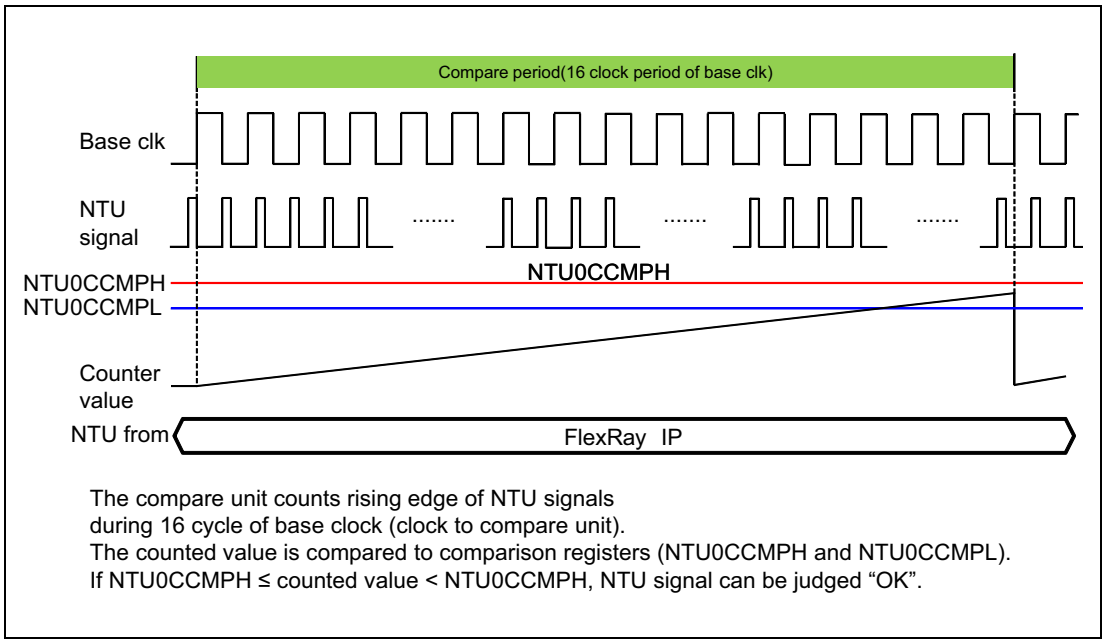
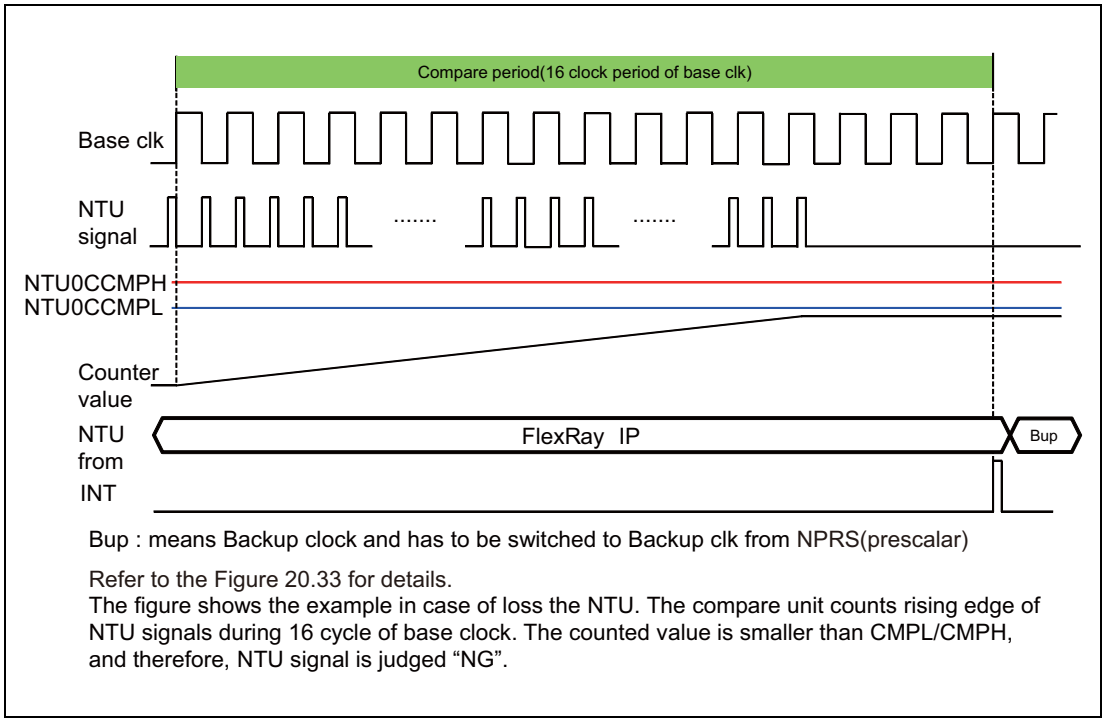


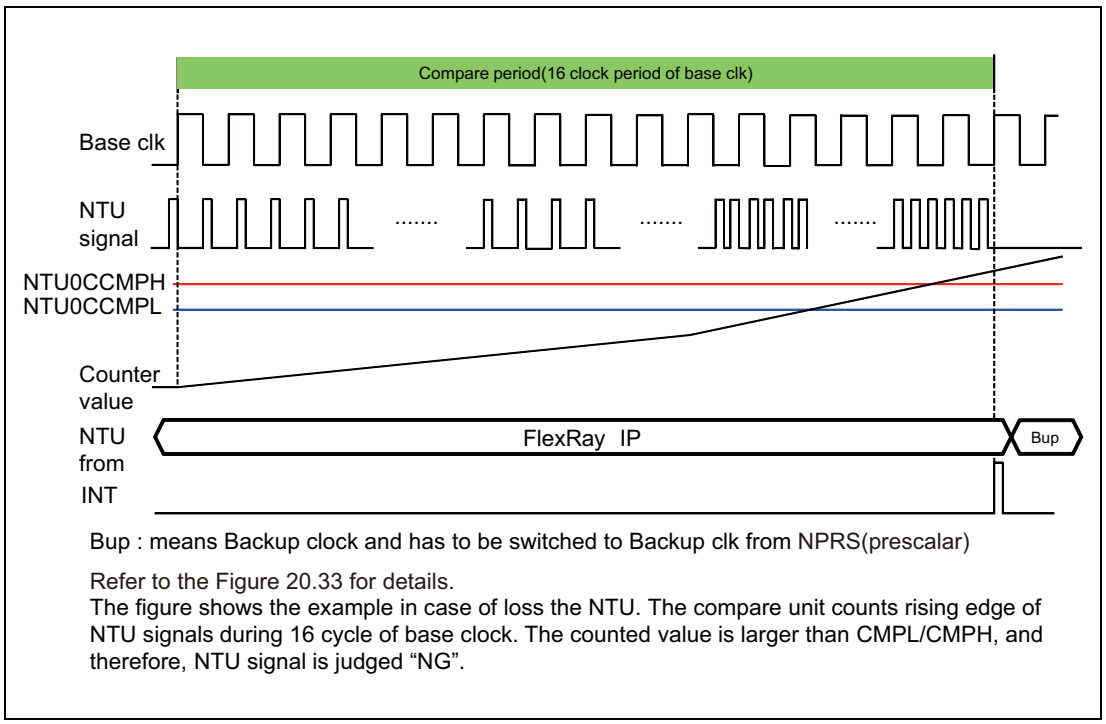

**Figure 20.34 The NTU loss monitor operation in compare unit (in case of OK)**




Bup : means Backup clock and has to be switched to Backup clk from NPRS(prescalar)

Refer to the Figure 20.33 for details.
The figure shows the example in case of loss the NTU. The compare unit counts rising edge of NTU signals during 16 cycle of base clock. The counted value is smaller than CMPL/CMPH, and therefore, NTU signal is judged “NG”.

**Figure 20.35 The NTU loss monitor operation in compare unit (in case of loss 1)**




Bup : means Backup clock and has to be switched to Backup clk from NPRS(prescalar)

Refer to the Figure 20.33 for details.
The figure shows the example in case of loss the NTU. The compare unit counts rising edge of NTU signals during 16 cycle of base clock. The counted value is larger than CMPL/CMPH, and therefore, NTU signal is judged “NG”.

**Figure 20.36 The NTU loss monitor operation in compare unit (in case of loss 2)**

### 20.3.19 Port sharing between FlexRay IP and RLIN

FlexRay 0 channel A pins will be shared with RLIN30 module.

In detail, see **Section 2, Pin Functions**.

### 20.3.20 Wakeup capability

To provide a wake up capability in the FlexRay IP, detecting a falling edge of bus signal can be used as a trigger for interrupt controller. After receiving this trigger, the interrupt controller will be able to run SW to wake up the FlexRay IP.

For this functionality it is necessary to connect FLXnRXDA and FLXAnRXDB pins (n = 0, 1) with an external interrupt input (INTP), therefore port sharing between INTP and FLXnRXDA and FLXAnRXDB pins (n = 0, 1) of FlexRay IP will be implemented.

### 20.3.21 Limited Reset and Module stand-by

The FlexRay can be reset by limited reset from SYSCTRL. The Limited Reset is functionally equivalent to a hardware reset. Software must ensure that FlexRay is halted. (See **Section 20.3.1.2, FlexRay Module Disable)**

The FlexRay clock can be disabled by the SYSCTRL module stand-by function. Software must ensure that FlexRay is halted if module stand-by enable.

## 20.4 CAUTION

| No. | Outline |
|---|---|
| ID1 | A sequence of received WUS may generate redundant FLXAnFRSIR.WUPA/B events. |
| ID2 | Erroneous cycle offset during startup after abort of startup or normal operation by a READY or FREEZE command |
| ID3 | First WUS following received valid WUP may be ignored. |
| ID4 | READY command accepted in READY state. |
| ID5 | Slot Status vPOC!SlotMode is reset immediately when entering HALT state (CCSV.SLM[1:0] = "00"). |

| | | |
|---|---|---|
| ID1 | | A sequence of received WUS may generate redundant FLXAnFRSIR.WUPA/B events. |
| | Description | If a sequence of wakeup symbols (WUS) is received, all separated by appropriate idle phases, a valid wakeup pattern (WUP) should be detected after every second WUS. The FlexRay IP detects a valid wakeup pattern after the second WUS and then after each following WUS. |
| | Scope | The erratum is limited to the case where the application program frequently resets the appropriate FLXAnFRSIR.WUPA/B bits |
| | Effects | In the described case there are more FLXAnFRSIR.WUPA/B events seen than expected |
| | Workaround | Ignore redundant FLXAnFRSIR.WUPA/B events |
| ID2 | | Erroneous cycle offset during startup after abort of startup or normal operation by a READY or FREEZE command |
| | Description | An abort of startup or normal operation by a READY or FREEZE command near the macrotick border may lead to the effect that the state INITIALIZE_SCHEDULE is one macrotick too short during the first following integration attempt. This leads to an early cycle start in state INTEGRATION_COLDSTART_CHECK or INTEGRATION_CONSISTENCY_CHECK.<br>As a result the integrating node calculates a cycle offset of one macrotick at the end of the first even/odd cycle pair in the states INTEGRATION_COLDSTART_CHECK or INTEGRATION_CONSISTENCY_CHECK and tries to correct this offset.<br>If the node is able to correct the offset of one macrotick (pOffsetCorrectionOut >> gdMacrotick), the node enters NORMAL_ACTIVE with the first startup attempt.<br>If the node is not able to correct the offset error because pOffsetCorrectionOut is too small (pOffsetCorrectionOut ≤ gdMacrotick), the node enters ABORT_STARTUP and is ready to try startup again. The next (second) startup attempt is not affected by this erratum. |
| | Scope | The erratum is limited to applications where READY or FREEZE command is used to leave STARTUP, NORMAL_ACTIVE, or NORMAL_PASSIVE state |
| | Effects | In the described case the integrating node tries to correct an erroneous cycle offset of one macrotick during startup. |
| | Workaround | With a configuration of pOffsetCorrectionOut >> gdMacrotick • (1+cClockDeviationMax)<br>the node will be able to correct the offset and therefore also be able to successfully integrate. |
| ID3 | | First WUS following received valid WUP may be ignored. |
| | Description | When the protocol engine is in state WAKEUP_LISTEN and receives a valid wakeup pattern (WUP), it transfers into state READY and updates the wakeup status vector FLXAnFRCCSV.WSV[2:0] as well as the status interrupt flags SIR.WST and FLXAnFRSIR.WUPA/B. If the received wakeup pattern continues, the protocol engine may ignore the first wakeup symbol (WUS) following the state transition and signals the next FLXAnFRSIR.WUPA/B at the third instead of the second WUS. |
| | Scope | The erratum is limited to the reception of redundant wakeup patterns. |
| | Effects | Delayed setting of status interrupt flags FLXAnFRSIR.WUPA/B for redundant wakeup patterns |
| | Workaround | None needed. |

| | | |
|---|---|---|
| ID4 | | READY command accepted in READY state. |
| | Description | The FlexRay module does not ignore a READY command while in READY state. |
| | Scope | The erratum is limited to the READY state. |
| | Effects | Flag FLXAnFRCCSV.CSI is set. Cold starting needs to be enabled by POC command ALLOW_COLDSTART (FLXAnFRSUCC1.CMD = "1001"). |
| | Workaround | None needed. |
| ID5 | Description | When the protocol engine is in the states NORMAL_ACTIVE or NORMAL_PASSIVE, a HALT or FREEZE command issued by the Host resets vPOC!SlotMode immediately to SINGLE slot mode (FLXAnFRCCSV.SLM[1:0] = "00"). According to the FlexRay protocol specification, the slot mode should not be reset to SINGLE slot mode before the following state transition from HALT to DEFAULT_CONFIG state. |
| | Scope | The erratum is limited to the HALT state |
| | Effects | The slot status vPOC!SlotMode is reset to SINGLE when entering HALT state. |
| | Workaround | None needed. |

## 20.5 Difference among P1M-C, P1H-C and P1H-CE

| Instance Name | RH850/P1x-C | | | |
|---|---|---|---|---|
| | P1M-C, P1H-C (4MB, BGA-156) | P1H-C (4MB, BGA-292) | P1H-C (8MB) | P1H-CE |
| FLXA0 | √ | √ | √ | √ |
| FLXA1 | — | √ | √ | √ |

# Section 21 Ethernet Controller (ETNA)

This section describes the Ethernet Controller (ETNA) module.

The first subsection describes all the characteristics specific to the RH850/P1x-C such as channels, register base addresses, and input/output signal names.
The second and subsequent subsections describe characteristics of Ethernet.

## 21.1 Feature Overview of RH850/P1x-C Ethernet Controller

### 21.1.1 Number of Channels and Ports

The RH850/P1x-C includes one physical channel. This physical channel serves two MAC layer interface ports (hereafter referred to as port 0 and port 1), both of which can be made to perform transmission and reception independently.

**Table 21.1 Ethernet Channels**

| Ethernet | P1M-C, P1H-C (4MB, BGA-156) | P1H-C (4MB, BGA-292) and P1H-CE |
|---|---|---|
| Number of Channels | 1 | 1 |
| Name | ETNA0 | ETNA0 |
| Number of Ports per Channel | 1 | 2 |

#### Index n and m

This section identifies each Ethernet channel by “n” (n = 0) and port by m (m = 0, 1). For example, the E-DMAC Start Register is described as ETNAnEDSRm.

### 21.1.2 Register Addresses

Ethernet register addresses are represented by an offset from the base address <ETNAn_base>.

The following table shows the base address <ETNAn_base> of each Ethernet module.

**Table 21.2 Register Base Address <ETNAn_base>**

| Ethernetn Channel | <ETNAn_base> Address |
|---|---|
| ETNA0 | $1002\ 4000_H$ |

### 21.1.3 Clock Supply

The following clock input is supplied for the Ethernet module.

**Table 21.3 Ethernet Clock Source**

| Clock | explanation | specification |
|---|---|---|
| HCLK_0 | H-bus interface clock*1 | CLK_HSB |
| HCLK_1 | H-bus interface clock*1 | CLK_HSB |
| ETH0TXCLK | MII TX CLK for ETNA0 port 0 | External clock |
| ETH1TXCLK | MII TX CLK for ETNA0 port 1 | External clock |
| ETH0RXCLK | MII RX CLK for ETNA0 port 0 | External clock |
| ETH1RXCLK | MII RX CLK for ETNA0 port 1 | External clock |
| ETH0REF50CK | RMII reference clock for ETNA0 port 0 | External clock |
| ETH1REF50CK | RMII reference clock for ETNA0 port 1 | External clock |

Note 1. The input frequnecy must be in the rage of 60MHz to 80MHz

### 21.1.4 Interrupts and DMA/DTS

The Ethernet module can generate the following interrupt requests.

**Table 21.4 Interrupt Requests**

| Unit Interrupt Name | Outline | Interrupt Number | DMA Trigger Number | DTS Trigger Number |
|---|---|---|---|---|
| **ETNA0** | | | | |
| INTETNA0 | Ethernet interrupt for Port 0 (HINT_0) | 184 | — | — |
| INTETNA1 | Ethernet interrupt for Port 1 (HINT_1) | 185 | — | — |

### 21.1.5 Difference table of P1M-C, P1H-C, P1H-CE product

Table 21.5 Difference of number of macro

| Macro | Device | | | | |
|---|---|---|---|---|---|
| | P1M-C | P1H-C (4MB, BGA-156) | P1H-C (4MB, BGA-292) | P1H-C (8MB) | P1H-CE |
| ETNA0 port 0 | 1 | 1 | 1 | 1 | 1 |
| ETNA0 port 1 | — | — | 1 | 1 | 1 |

### 21.1.6 External Input/Output Signals

The following table shows the Ethernet input/output signals.

Table 21.6 Ethernet External Input / Output Signals

| Pin Name | I/O | Function | MII | RMII |
|---|---|---|---|---|
| ETHmTXCLK | Input | Transmission clock | √ | |
| ETHmRXCLK | Input | Reception clock | √ | |
| ETHmREF50CK | Input | RMII reference clock | | √ |
| ETHmTXEN | Output | Transmission data enable | √ | √ |
| ETHmTXD3 | Output | Transmission data 3 | √ | |
| ETHmTXD2 | Output | Transmission data 2 | √ | |
| ETHmTXD1 | Output | Transmission data 1 | √ | √ |
| ETHmTXD0 | Output | Transmission data 0 | √ | √ |
| ETHmTXER | Output | Transmission data error | √ | |
| ETHmRXDV | Input | Reception data valid | √ | |
| ETHmRXD3 | Input | Reception data 3 | √ | |
| ETHmRXD2 | Input | Reception data 2 | √ | |
| ETHmRXD1 | Input | Reception data 1 | √ | √ |
| ETHmRXD0 | Input | Reception data 0 | √ | √ |
| ETHmRXER | Input | Reception data error | √ | √ |
| ETHmCRS | Input | Carrier detection signal | v | √ |
| ETHmCOL | Input | Collision detection signal | √ | |
| ETHmMDC | Output | Serial management interface transfer clock | √ | √ |
| ETHmMDIO | Input/ Output | Serial management interface data input / output<br>Input pin ETHmMDI<br>Output pin ETHmMDO | √ | √ |
| ETHmLINKSTA | Input | Link status from PHY | √ | √ |
| ETHmWOL | Output | Wake on LAN signal (magic packet detection) | √ | √ |

Note 1. In this device, ETHmTXER is unused and fixed to low.

Note 2. The minimum pulse width of ETHmRXER in RMII mode is depend on transfer speed mode. When operating mode is RMII mode with 10Mbps transfer speed, the minimum pulse of ETHmRXER need to be at least 1 clock of ETHmREF50CK (200ns) against RMII standard.

Note 3. ETH0LINKSTA pin is not supported by P1M-C (BGA-156) and P1H-C (4MB, BGA-156).

## 21.2 Overview

The Ethernet Controller (ETNA) conforms to the IEEE802.3 MAC (Media Access Control) layer standard. Connecting to an IEEE802.3 compliant physical layer device (PHY) enables the ETNA to transmit and receive Ethernet frames. The ETNA is equipped with MII (Media Independent Interface) and RMII (Reduced Media Independent Interface) and connected to the Ethernet Direct Memory Access Controller (EDMAC), which carries out high-speed data transfer (10M/100Mbps) to and from the system memory.

### 21.2.1 Functional Overview

**Table 21.7 Outline of Ethernet function**

| Item | Specification |
|---|---|
| MAC (Media Access Controller) | • 100Mbps or 10Mbps data transfer<br>• Full or half-duplex mode<br>• PHY interface: MII, RMII |
| MAC port number | Implements 2 ports |
| DMA transfer function | • Each port implements the AHB master function and transfers the received/transmitted data from/to the RAM memory area.<br>• Supports the 8 times of the 32-bit burst transfer.<br>• Descriptor management system used by EDMAC (Data transfer without CPU intervention)<br>• Single-frame/single-descriptor and single-frame/multi-descriptor operation.<br>• Transmission/reception status indication in descriptor. |
| Address filter function | • Performs address filtering based on the destination address.<br>• Up to 32 different addresses can be configured individually.<br>• Supports the storm filter for broadcast frames.<br>• Supports the mode enable/disable function for broadcast frames by using CAM entry table. |
| Others | Supports following features:<br>• Magic packet detection and Wake-On-LAN signal output<br>• Flow conforms to IEEE802.3<br>• PAUSE control frame format<br>• CRC calculation for RX/TX frames<br>• Configurable address filtering function (CAM function).<br>• Integrated transmit/receive FIFO (TX = 2 KB, RX = 2 KB)<br>• Interrupts for configurable error and status conditions<br>• Internal loopback function (external loopback function can be realized with PHY interface)<br>• MAC management counters |

### 21.2.2 Block Diagram

The following **Figure 21.1** shows a block diagram of the Ethernet module.

The ETNA can transfer the transmitted or received Ethernet frame data to and from the transmit/ receive buffer in the memory at high speed using a dedicated direct memory access controller (E-DMAC).

The ETNA also has an on-chip TSU (Transfer Switching Unit) for CAM processing.

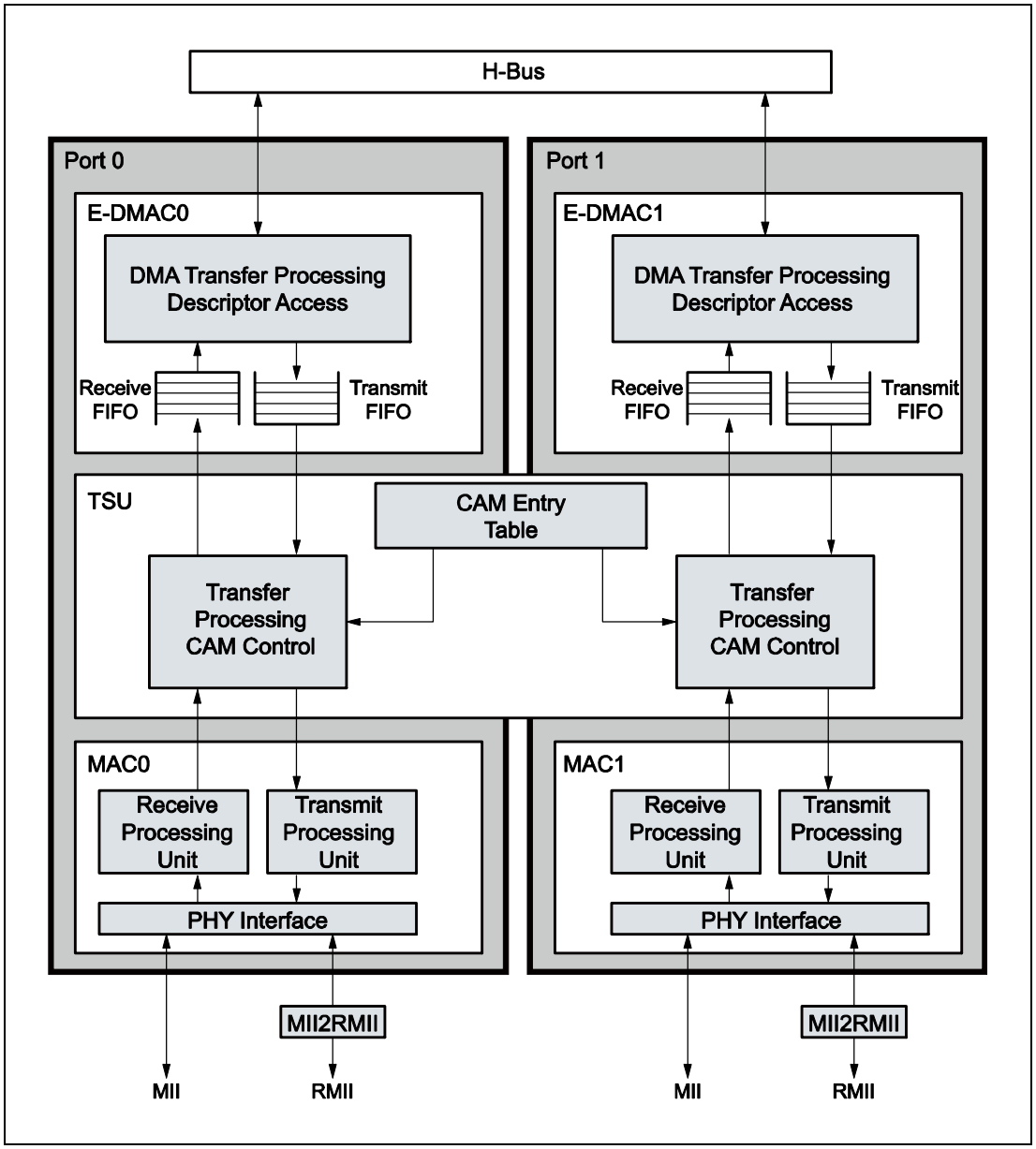



Figure 21.1 Block Diagram of Ethernet

# 21.3 Registers

## 21.3.1 Overview of Register

Ethernet is controlled and operated by the following registers.

**Table 21.8 Overview of Ethernet Registers (1/8)**

| Description | Register Name | Access Width | Address | Access Protection | |
|---|---|---|---|---|---|
| | | | | HBG | Other |
| E-DMAC Start Register 0 | ETNAnEDSR0 | 32 | <ETNAn_base> + $0000_H$ | HBG.Ethernet n*1 | — |
| Transmit Descriptor List Start Address Register 0 | ETNAnTDLAR0 | 32 | <ETNAn_base> + $0010_H$ | HBG.Ethernet n*1 | — |
| Transmit Descriptor Fetch Address Register 0 | ETNAnTDFAR0 | 32 | <ETNAn_base> + $0014_H$ | HBG.Ethernet n*1 | — |
| Transmit Descriptor Finished Address Register 0 | ETNAnTDFXR0 | 32 | <ETNAn_base> + $0018_H$ | HBG.Ethernet n*1 | — |
| Transmit Descriptor Final Flag Register 0 | ETNAnTDFFR0 | 32 | <ETNAn_base> + $001C_H$ | HBG.Ethernet n*1 | — |
| Receive Descriptor List Start Address Register 0 | ETNAnRDLAR0 | 32 | <ETNAn_base> + $0030_H$ | HBG.Ethernet n*1 | — |
| Receive Descriptor Fetch Address Register 0 | ETNAnRDFAR0 | 32 | <ETNAn_base> + $0034_H$ | HBG.Ethernet n*1 | — |
| Receive Descriptor Finished Address Register 0 | ETNAnRDFXR0 | 32 | <ETNAn_base> + $0038_H$ | HBG.Ethernet n*1 | — |
| Receive Descriptor Final Flag Register 0 | ETNAnRDFFR0 | 32 | <ETNAn_base> + $003C_H$ | HBG.Ethernet n*1 | — |
| E-DMAC Mode Register 0 | ETNAnEDMR0 | 32 | <ETNAn_base> + $0400_H$ | HBG.Ethernet n*1 | — |
| E-DMAC Transmit Request Register 0 | ETNAnEDTRR0 | 32 | <ETNAn_base> + $0408_H$ | HBG.Ethernet n*1 | — |
| E-DMAC Receive Request Register 0 | ETNAnEDRRR0 | 32 | <ETNAn_base> + $0410_H$ | HBG.Ethernet n*1 | — |
| E-MAC/E-DMAC Status Register 0 | ETNAnEESR0 | 32 | <ETNAn_base> + $0428_H$ | HBG.Ethernet n*1 | — |
| E-MAC/E-DMAC Status Interrupt Permission Register 0 | ETNAnEESIPR0 | 32 | <ETNAn_base> + $0430_H$ | HBG.Ethernet n*1 | — |
| Transmit/Receive Status Copy Enable Register 0 | ETNAnTRSCER0 | 32 | <ETNAn_base> + $0438_H$ | HBG.Ethernet n*1 | — |
| Receive Missed-Frame Counter Register 0 | ETNAnRMFCR0 | 32 | <ETNAn_base> + $0440_H$ | HBG.Ethernet n*1 | — |
| FIFO Depth Register 0 | ETNAnFDR0 | 32 | <ETNAn_base> + $0450_H$ | HBG.Ethernet n*1 | — |
| Receiving Method Control Register 0 | ETNAnRMCR0 | 32 | <ETNAn_base> + $0458_H$ | HBG.Ethernet n*1 | — |
| Receive Data Padding Insert Register 0 | ETNAnRPADIR0 | 32 | <ETNAn_base> + $0460_H$ | HBG.Ethernet n*1 | — |
| Overflow Alert FIFO Threshold Register 0 | ETNAnFCFTR0 | 32 | <ETNAn_base> + $0468_H$ | HBG.Ethernet n*1 | — |
| Transmit FIFO Status Register 0 | ETNAnTFSR0 | 32 | <ETNAn_base> + $0480_H$ | HBG.Ethernet n*1 | — |
| Receive FIFO Status Register 0 | ETNAnRFSR0 | 32 | <ETNAn_base> + $0488_H$ | HBG.Ethernet n*1 | — |

Table 21.8 Overview of Ethernet Registers (2/8)

| Description | Register Name | Access Width | Address | Access Protection | |
|---|---|---|---|---|---|
| | | | | HBG | Other |
| E-MAC Mode Register 0 | ETNAnECMR0 | 32 | <ETNAn_base> + $0500_H$ | HBG.Ethernet n*[1] | — |
| Receive Frame Length Register 0 | ETNAnRFLR0 | 32 | <ETNAn_base> + $0508_H$ | HBG.Ethernet n*[1] | — |
| E-MAC Status Register 0 | ETNAnECSR0 | 32 | <ETNAn_base> + $0510_H$ | HBG.Ethernet n*[1] | — |
| E-MAC Interrupt Permission Register 0 | ETNAnECSIPR0 | 32 | <ETNAn_base> + $0518_H$ | HBG.Ethernet n*[1] | — |
| PHY Interface Register 0 | ETNAnPIR0 | 32 | <ETNAn_base> + $0520_H$ | HBG.Ethernet n*[1] | — |
| PHY link status Register 0 | ETNAnPLSR0 | 32 | <ETNAn_base> + $0528_H$ | HBG.Ethernet n*[1] | — |
| PHY interrupt polarity register0 | ETNAnPIPR0 | 32 | <ETNAn_base> + $052C_H$ | HBG.Ethernet n*[1] | — |
| Automatic PAUSE Frame Register 0 | ETNAnAPR0 | 32 | <ETNAn_base> + $0554_H$ | HBG.Ethernet n*[1] | — |
| Manual PAUSE Frame Register 0 | ETNAnMPR0 | 32 | <ETNAn_base> + $0558_H$ | HBG.Ethernet n*[1] | — |
| PAUSE Frame Transmit Counter Register 0 | ETNAnPFTCR0 | 32 | <ETNAn_base> + $055C_H$ | HBG.Ethernet n*[1] | — |
| PAUSE Frame Receive Counter Register | ETNAnPFRCR0 | 32 | <ETNAn_base> + $0560_H$ | HBG.Ethernet n*[1] | — |
| Automatic PAUSE Frame Retransmit Count Register 0 | ETNAnTPAUSER0 | 32 | <ETNAn_base> + $0564_H$ | HBG.Ethernet n*[1] | — |
| Automatic PAUSE Frame Retransmit counter Register0 | ETNAnTPFRCR0 | 32 | <ETNAn_base> + $0568_H$ | HBG.Ethernet n*[1] | — |
| Broadcast Frame Receive Count Set Register 0 | ETNAnBCFRR0 | 32 | <ETNAn_base> + $056C_H$ | HBG.Ethernet n*[1] | — |
| Count Reset of BCFRR 0 | ETNAnCRBCFRR0 | 32 | <ETNAn_base> + $0570_H$ | HBG.Ethernet n*[1] | — |
| ETNA Mode Register 0 | ETNAnEMR0 | 32 | <ETNAn_base> + $05B0_H$ | HBG.Ethernet n*[1] | — |
| MAC Address High Register 0 | ETNAnMAHR0 | 32 | <ETNAn_base> + $05C0_H$ | HBG.Ethernet n*[1] | — |
| MAC Address Low Register 0 | ETNAnMALR0 | 32 | <ETNAn_base> + $05C8_H$ | HBG.Ethernet n*[1] | — |
| Transmit Retry Over Counter Register 0 | ETNAnTROCR0 | 32 | <ETNAn_base> + $0700_H$ | HBG.Ethernet n*[1] | — |
| Delayed Collision Detect Counter Register 0 | ETNAnCDCR0 | 32 | <ETNAn_base> + $0708_H$ | HBG.Ethernet n*[1] | — |
| Lost Carrier Counter Register 0 | ETNAnLCCR0 | 32 | <ETNAn_base> + $0710_H$ | HBG.Ethernet n*[1] | — |
| CRC Error Frame Receive Counter Register 0 | ETNAnCEFCR0 | 32 | <ETNAn_base> + $0740_H$ | HBG.Ethernet n*[1] | — |
| Frame Receive Error Counter Register 0 | ETNAnFRECR0 | 32 | <ETNAn_base> + $0748_H$ | HBG.Ethernet n*[1] | — |
| Too-Short Frame Receive Counter Register 0 | ETNAnTSFRCR0 | 32 | <ETNAn_base> + $0750_H$ | HBG.Ethernet n*[1] | — |
| Too-Long Frame Receive Counter Register 0 | ETNAnTLFRCR0 | 32 | <ETNAn_base> + $0758_H$ | HBG.Ethernet n*[1] | — |

Table 21.8 Overview of Ethernet Registers (3/8)

| Description | Register Name | Access Width | Address | Access Protection | |
|---|---|---|---|---|---|
| | | | | HBG | Other |
| Residual-Bit Frame Receive Counter Register 0 | ETNAnRFCR0 | 32 | <ETNAn_base> + $0760_H$ | HBG.Ethernet n*[1] | — |
| Multicast Address Frame Receive Counter Register 0 | ETNAnMAFCR0 | 32 | <ETNAn_base> + $0778_H$ | HBG.Ethernet n*[1] | — |
| Software Reset Register | ETNAnARSTR | 32 | <ETNAn_base> + $0800_H$ | HBG.Ethernet n*[1] | — |
| TSU Counter Reset Register | ETNAnTSU_CTRST | 32 | <ETNAn_base> + $0804_H$ | HBG.Ethernet n*[1] | — |
| Receive Function Set Register | ETNAnTSU_FWSL0 | 32 | <ETNAn_base> + $0830_H$ | HBG.Ethernet n*[1] | — |
| Receive Function Set Register | ETNAnTSU_FWSL1 | 32 | <ETNAn_base> + $0834_H$ | HBG.Ethernet n*[1] | — |
| TSU Function Set Register | ETNAnTSU_FWSLC | 32 | <ETNAn_base> + $0838_H$ | HBG.Ethernet n*[1] | — |
| TSU Status Register | ETNAnTSU_FWSR | 32 | <ETNAn_base> + $0850_H$ | HBG.Ethernet n*[1] | — |
| TSU Status interrupt mask register | ETNAnTSU_FWINMK | 32 | <ETNAn_base> + $0854_H$ | HBG.Ethernet n*[1] | — |
| VLANtag Set Register (Port 0) | ETNAnTSU_VTAG0 | 32 | <ETNAn_base> + $0858_H$ | HBG.Ethernet n*[1] | — |
| VLANtag Set Register (Port 1) | ETNAnTSU_VTAG1 | 32 | <ETNAn_base> + $085C_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table Busy Register | ETNAnTSU_ADSBSY | 32 | <ETNAn_base> + $0860_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table Enable Register | ETNAnTSU_TEN | 32 | <ETNAn_base> + $0864_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table POST1 Register | ETNAnTSU_POST1 | 32 | <ETNAn_base> + $0870_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table POST2 Register | ETNAnTSU_POST2 | 32 | <ETNAn_base> + $0874_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table POST3 Register | ETNAnTSU_POST3 | 32 | <ETNAn_base> + $0878_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table POST4 Register | ETNAnTSU_POST4 | 32 | <ETNAn_base> + $087C_H$ | HBG.Ethernet n*[1] | — |
| Transmit Frame Counter Register (Port 0) (Normal Transmission Only) | ETNAnTXNLCR0 | 32 | <ETNAn_base> + $0880_H$ | HBG.Ethernet n*[1] | — |
| Transmit Frame Counter Register (Port 0) (Normal and Erroneous Transmission) | ETNAnTXALCR0 | 32 | <ETNAn_base> + $0884_H$ | HBG.Ethernet n*[1] | — |
| Receive Frame Counter Register (Port 0) (Normal Reception Only) | ETNAnRXNLCR0 | 32 | <ETNAn_base> + $0888_H$ | HBG.Ethernet n*[1] | — |
| Receive Frame Counter Register (Port 0) (Normal and Erroneous Reception) | ETNAnRXALCR0 | 32 | <ETNAn_base> + $088C_H$ | HBG.Ethernet n*[1] | — |
| Transmit Frame Counter Register (Port 1) (Normal Transmission Only) | ETNAnTXNLCR1 | 32 | <ETNAn_base> + $08A0_H$ | HBG.Ethernet n*[1] | — |
| Transmit Frame Counter Register (Port 1) (Normal and Erroneous Transmission) | ETNAnTXALCR1 | 32 | <ETNAn_base> + $08A4_H$ | HBG.Ethernet n*[1] | — |
| Receive Frame Counter Register (Port 1) (Normal Reception Only) | ETNAnRXNLCR1 | 32 | <ETNAn_base> + $08A8_H$ | HBG.Ethernet n*[1] | — |
| Receive Frame Counter Register (Port 1) (Normal and Erroneous Reception) | ETNAnRXALCR1 | 32 | <ETNAn_base> + $08AC_H$ | HBG.Ethernet n*[1] | — |

Table 21.8 Overview of Ethernet Registers (4/8)

| Description | Register Name | Access Width | Address | Access Protection | |
|---|---|---|---|---|---|
| | | | | HBG | Other |
| CAM Entry Table 0H Registers | ETNAnTSU_ADRH0 | 32 | <ETNAn_base> + $0900_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 0L Registers | ETNAnTSU_ADRL0 | 32 | <ETNAn_base> + $0904_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 1H Registers | ETNAnTSU_ADRH1 | 32 | <ETNAn_base> + $0908_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 1L Registers | ETNAnTSU_ADRL1 | 32 | <ETNAn_base> + $090C_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 2H Registers | ETNAnTSU_ADRH2 | 32 | <ETNAn_base> + $0910_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 2L Registers | ETNAnTSU_ADRL2 | 32 | <ETNAn_base> + $0914_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 3H Registers | ETNAnTSU_ADRH3 | 32 | <ETNAn_base> + $0918_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 3L Registers | ETNAnTSU_ADRL3 | 32 | <ETNAn_base> + $091C_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 4H Registers | ETNAnTSU_ADRH4 | 32 | <ETNAn_base> + $0920_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 4L Registers | ETNAnTSU_ADRL4 | 32 | <ETNAn_base> + $0924_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 5H Registers | ETNAnTSU_ADRH5 | 32 | <ETNAn_base> + $0928_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 5L Registers | ETNAnTSU_ADRL5 | 32 | <ETNAn_base> + $092C_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 6H Registers | ETNAnTSU_ADRH6 | 32 | <ETNAn_base> + $0930_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 6L Registers | ETNAnTSU_ADRL6 | 32 | <ETNAn_base> + $0934_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 7H Registers | ETNAnTSU_ADRH7 | 32 | <ETNAn_base> + $0938_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 7L Registers | ETNAnTSU_ADRL7 | 32 | <ETNAn_base> + $093C_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 8H Registers | ETNAnTSU_ADRH8 | 32 | <ETNAn_base> + $0940_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 8L Registers | ETNAnTSU_ADRL8 | 32 | <ETNAn_base> + $0944_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 9H Registers | ETNAnTSU_ADRH9 | 32 | <ETNAn_base> + $0948_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 9L Registers | ETNAnTSU_ADRL9 | 32 | <ETNAn_base> + $094C_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 10H Registers | ETNAnTSU_ADRH10 | 32 | <ETNAn_base> + $0950_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 10L Registers | ETNAnTSU_ADRL10 | 32 | <ETNAn_base> + $0954_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 11H Registers | ETNAnTSU_ADRH11 | 32 | <ETNAn_base> + $0958_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 11L Registers | ETNAnTSU_ADRL11 | 32 | <ETNAn_base> + $095C_H$ | HBG.Ethernet n*1 | — |
| CAM Entry Table 12H Registers | ETNAnTSU_ADRH12 | 32 | <ETNAn_base> + $0960_H$ | HBG.Ethernet n*1 | — |

Table 21.8 Overview of Ethernet Registers (5/8)

| Description | Register Name | Access Width | Address | Access Protection | |
|---|---|---|---|---|---|
| | | | | HBG | Other |
| CAM Entry Table 12L Registers | ETNAnTSU_ADRL12 | 32 | <ETNAn_base> + $0964_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 13H Registers | ETNAnTSU_ADRH13 | 32 | <ETNAn_base> + $0968_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 13L Registers | ETNAnTSU_ADRL13 | 32 | <ETNAn_base> + $096C_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 14H Registers | ETNAnTSU_ADRH14 | 32 | <ETNAn_base> + $0970_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 14L Registers | ETNAnTSU_ADRL14 | 32 | <ETNAn_base> + $0974_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 15H Registers | ETNAnTSU_ADRH15 | 32 | <ETNAn_base> + $0978_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 15L Registers | ETNAnTSU_ADRL15 | 32 | <ETNAn_base> + $097C_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 16H Registers | ETNAnTSU_ADRH16 | 32 | <ETNAn_base> + $0980_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 16L Registers | ETNAnTSU_ADRL16 | 32 | <ETNAn_base> + $0984_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 17H Registers | ETNAnTSU_ADRH17 | 32 | <ETNAn_base> + $0988_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 17L Registers | ETNAnTSU_ADRL17 | 32 | <ETNAn_base> + $098C_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 18H Registers | ETNAnTSU_ADRH18 | 32 | <ETNAn_base> + $0990_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 18L Registers | ETNAnTSU_ADRL18 | 32 | <ETNAn_base> + $0994_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 19H Registers | ETNAnTSU_ADRH19 | 32 | <ETNAn_base> + $0998_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 19L Registers | ETNAnTSU_ADRL19 | 32 | <ETNAn_base> + $099C_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 20H Registers | ETNAnTSU_ADRH20 | 32 | <ETNAn_base> + $09A0_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 20L Registers | ETNAnTSU_ADRL20 | 32 | <ETNAn_base> + $09A4_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 21H Registers | ETNAnTSU_ADRH21 | 32 | <ETNAn_base> + $09A8_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 21L Registers | ETNAnTSU_ADRL21 | 32 | <ETNAn_base> + $09AC_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 22H Registers | ETNAnTSU_ADRH22 | 32 | <ETNAn_base> + $09B0_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 22L Registers | ETNAnTSU_ADRL22 | 32 | <ETNAn_base> + $09B4_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 23H Registers | ETNAnTSU_ADRH23 | 32 | <ETNAn_base> + $09B8_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 23L Registers | ETNAnTSU_ADRL23 | 32 | <ETNAn_base> + $09BC_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 24H Registers | ETNAnTSU_ADRH24 | 32 | <ETNAn_base> + $09C0_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 24L Registers | ETNAnTSU_ADRL24 | 32 | <ETNAn_base> + $09C4_H$ | HBG.Ethernet n*[1] | — |

Table 21.8 Overview of Ethernet Registers (6/8)

| Description | Register Name | Access Width | Address | Access Protection | |
|---|---|---|---|---|---|
| | | | | HBG | Other |
| CAM Entry Table 25H Registers | ETNAnTSU_ADRH25 | 32 | <ETNAn_base> + $09C8_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 25L Registers | ETNAnTSU_ADRL25 | 32 | <ETNAn_base> + $09CC_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 26H Registers | ETNAnTSU_ADRH26 | 32 | <ETNAn_base> + $09D0_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 26L Registers | ETNAnTSU_ADRL26 | 32 | <ETNAn_base> + $09D4_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 27H Registers | ETNAnTSU_ADRH27 | 32 | <ETNAn_base> + $09D8_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 27L Registers | ETNAnTSU_ADRL27 | 32 | <ETNAn_base> + $09DC_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 28H Registers | ETNAnTSU_ADRH28 | 32 | <ETNAn_base> + $09E0_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 28L Registers | ETNAnTSU_ADRL28 | 32 | <ETNAn_base> + $09E4_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 29H Registers | ETNAnTSU_ADRH29 | 32 | <ETNAn_base> + $09E8_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 29L Registers | ETNAnTSU_ADRL29 | 32 | <ETNAn_base> + $09EC_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 30H Registers | ETNAnTSU_ADRH30 | 32 | <ETNAn_base> + $09F0_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 30L Registers | ETNAnTSU_ADRL30 | 32 | <ETNAn_base> + $09F4_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 31H Registers | ETNAnTSU_ADRH31 | 32 | <ETNAn_base> + $09F8_H$ | HBG.Ethernet n*[1] | — |
| CAM Entry Table 31L Registers | ETNAnTSU_ADRL31 | 32 | <ETNAn_base> + $09FC_H$ | HBG.Ethernet n*[1] | — |
| HDMAC Operation Mode Register 0 | ETNAnHDMMDR0 | 32 | <ETNAn_base> + $0C04_H$ | HBG.Ethernet n*[1] | — |
| HDMAC Interrupt Status Register 1 | ETNAnHDMISR0 | 32 | <ETNAn_base> + $0C10_H$ | HBG.Ethernet n*[1] | — |
| HDMAC Interrupt Enable Register 1 | ETNAnHDMIER0 | 32 | <ETNAn_base> + $0C14_H$ | HBG.Ethernet n*[1] | — |
| E-DMAC Start Register 1 | ETNAnEDSR1 | 32 | <ETNAn_base> + $1000_H$ | HBG.Ethernet n*[1] | — |
| Transmit Descriptor List Start Address Register 1 | ETNAnTDLAR1 | 32 | <ETNAn_base> + $1010_H$ | HBG.Ethernet n*[1] | — |
| Transmit Descriptor Fetch Address Register 1 | ETNAnTDFAR1 | 32 | <ETNAn_base> + $1014_H$ | HBG.Ethernet n*[1] | — |
| Transmit Descriptor Finished Address Register 1 | ETNAnTDFXR1 | 32 | <ETNAn_base> + $1018_H$ | HBG.Ethernet n*[1] | — |
| Transmit Descriptor Final Flag Register 1 | ETNAnTDFFR1 | 32 | <ETNAn_base> + $101C_H$ | HBG.Ethernet n*[1] | — |
| Receive Descriptor List Start Address Register 1 | ETNAnRDLAR1 | 32 | <ETNAn_base> + $1030_H$ | HBG.Ethernet n*[1] | — |
| Receive Descriptor Fetch Address Register 1 | ETNAnRDFAR1 | 32 | <ETNAn_base> + $1034_H$ | HBG.Ethernet n*[1] | — |
| Receive Descriptor Finished Address Register 1 | ETNAnRDFXR1 | 32 | <ETNAn_base> + $1038_H$ | HBG.Ethernet n*[1] | — |

Table 21.8 Overview of Ethernet Registers (7/8)

| Description | Register Name | Access Width | Address | Access Protection | |
|---|---|---|---|---|---|
| | | | | HBG | Other |
| Receive Descriptor Final Flag Register 1 | ETNAnRDFFR1 | 32 | <ETNAn_base> + $103C_H$ | HBG.Ethernet n*1 | — |
| E-DMAC Mode Register 1 | ETNAnEDMR1 | 32 | <ETNAn_base> + $1400_H$ | HBG.Ethernet n*1 | — |
| E-DMAC Transmit Request Register 1 | ETNAnEDTRR1 | 32 | <ETNAn_base> + $1408_H$ | HBG.Ethernet n*1 | — |
| E-DMAC Receive Request Register 1 | ETNAnEDRRR1 | 32 | <ETNAn_base> + $1410_H$ | HBG.Ethernet n*1 | — |
| E-MAC/E-DMAC Status Register 1 | ETNAnEESR1 | 32 | <ETNAn_base> + $1428_H$ | HBG.Ethernet n*1 | — |
| E-MAC/E-DMAC Status Interrupt Permission Register 1 | ETNAnEESIPR1 | 32 | <ETNAn_base> + $1430_H$ | HBG.Ethernet n*1 | — |
| Transmit/Receive Status Copy Enable Register 1 | ETNAnTRSCER1 | 32 | <ETNAn_base> + $1438_H$ | HBG.Ethernet n*1 | — |
| Receive Missed-Frame Counter Register 1 | ETNAnRMFCR1 | 32 | <ETNAn_base> + $1440_H$ | HBG.Ethernet n*1 | — |
| FIFO Depth Register 1 | ETNAnFDR1 | 32 | <ETNAn_base> + $1450_H$ | HBG.Ethernet n*1 | — |
| Receiving Method Control Register 1 | ETNAnRMCR1 | 32 | <ETNAn_base> + $1458_H$ | HBG.Ethernet n*1 | — |
| Receive Data Padding Insert Register n | ETNAnRPADIR1 | 32 | <ETNAn_base> + $1460_H$ | HBG.Ethernet n*1 | — |
| Overflow Alert FIFO Threshold Register 1 | ETNAnFCFTR1 | 32 | <ETNAn_base> + $1468_H$ | HBG.Ethernet n*1 | — |
| Transmit FIFO Status Register 1 | ETNAnTFSR1 | 32 | <ETNAn_base> + $1480_H$ | HBG.Ethernet n*1 | — |
| Receive FIFO Status Register 1 | ETNAnRFSR1 | 32 | <ETNAn_base> + $1488_H$ | HBG.Ethernet n*1 | — |
| E-MAC Mode Register 1 | ETNAnECMR1 | 32 | <ETNAn_base> + $1500_H$ | HBG.Ethernet n*1 | — |
| Receive Frame Length Register 0 | ETNAnRFLR1 | 32 | <ETNAn_base> + $1508_H$ | HBG.Ethernet n*1 | — |
| E-MAC Status Register 1 | ETNAnECSR1 | 32 | <ETNAn_base> + $1510_H$ | HBG.Ethernet n*1 | — |
| E-MAC Interrupt Permission Register 1 | ETNAnECSIPR1 | 32 | <ETNAn_base> + $1518_H$ | HBG.Ethernet n*1 | — |
| PHY Interface Register 1 | ETNAnPIR1 | 32 | <ETNAn_base> + $1520_H$ | HBG.Ethernet n*1 | — |
| PHY link status Register 1 | ETNAnPLSR1 | 32 | <ETNAn_base> + $1528_H$ | HBG.Ethernet n*1 | — |
| PHY interrupt polarity register1 | ETNAnPIPR1 | 32 | <ETNAn_base> + 152C | HBG.Ethernet n*1 | — |
| Automatic PAUSE Frame Register 1 | ETNAnAPR1 | 32 | <ETNAn_base> + $1554_H$ | HBG.Ethernet n*1 | — |
| Manual PAUSE Frame Register 1 | ETNAnMPR1 | 32 | <ETNAn_base> + $1558_H$ | HBG.Ethernet n*1 | — |
| PAUSE Frame Transmit Counter Register 1 | ETNAnPFTCR1 | 32 | <ETNAn_base> + $155C_H$ | HBG.Ethernet n*1 | — |
| PAUSE Frame Receive Counter Register 1 | ETNAnPFRCR1 | 32 | <ETNAn_base> + $1560_H$ | HBG.Ethernet n*1 | — |

Table 21.8 Overview of Ethernet Registers (8/8)

| Description | Register Name | Access Width | Address | Access Protection | |
|---|---|---|---|---|---|
| | | | | HBG | Other |
| Automatic PAUSE Frame Retransmit Count Register 1 | ETNAnTPAUSER1 | 32 | <ETNAn_base> + $1564_H$ | HBG.Ethernet n*1 | — |
| Automatic PAUSE Frame Retransmit Counter Register 1 | ETNAnTPFRCR1 | 32 | <ETNAn_base> + $1568_H$ | HBG.Ethernet n*1 | — |
| Broadcast Frame Receive Count Set Register 1 | ETNAnBCFRR1 | 32 | <ETNAn_base> + $156C_H$ | HBG.Ethernet n*1 | — |
| Count Reset of BCFRR 1 | ETNAnCRBCFRR1 | 32 | <ETNAn_base> + $1570_H$ | HBG.Ethernet n*1 | — |
| ETNA Mode Register 1 | ETNAnEMR1 | 32 | <ETNAn_base> + $15B0_H$ | HBG.Ethernet n*1 | — |
| MAC Address High Register 1 | ETNAnMAHR1 | 32 | <ETNAn_base> + $15C0_H$ | HBG.Ethernet n*1 | — |
| MAC Address Low Register 1 | ETNAnMALR1 | 32 | <ETNAn_base> + $15C8_H$ | HBG.Ethernet n*1 | — |
| Transmit Retry Over Counter Register 1 | ETNAnTROCR1 | 32 | <ETNAn_base> + $1700_H$ | HBG.Ethernet n*1 | — |
| Delayed Collision Detect Counter Register 1 | ETNAnCDCR1 | 32 | <ETNAn_base> + $1708_H$ | HBG.Ethernet n*1 | — |
| Lost Carrier Counter Register 1 | ETNAnLCCR1 | 32 | <ETNAn_base> + $1710_H$ | HBG.Ethernet n*1 | — |
| CRC Error Frame Receive Counter Register 1 | ETNAnCEFCR1 | 32 | <ETNAn_base> + $1740_H$ | HBG.Ethernet n*1 | — |
| Frame Receive Error Counter Register 1 | ETNAnFRECR1 | 32 | <ETNAn_base> + $1748_H$ | HBG.Ethernet n*1 | — |
| Too-Short Frame Receive Counter Register 1 | ETNAnTSFRCR1 | 32 | <ETNAn_base> + $1750_H$ | HBG.Ethernet n*1 | — |
| Too-Long Frame Receive Counter Register 1 | ETNAnTLFRCR1 | 32 | <ETNAn_base> + $1758_H$ | HBG.Ethernet n*1 | — |
| Residual-Bit Frame Receive Counter Register 1 | ETNAnRFCR1 | 32 | <ETNAn_base> + $1760_H$ | HBG.Ethernet n*1 | — |
| Multicast Address Frame Receive Counter Register 0 | ETNAnMAFCR1 | 32 | <ETNAn_base> + $1778_H$ | HBG.Ethernet n*1 | — |
| HDMAC Operation Mode Register 1 | ETNAnHDMMDR1 | 32 | <ETNAn_base> + $1C04_H$ | HBG.Ethernet n*1 | — |
| HDMAC Interrupt Status Register 1 | ETNAnHDMISR1 | 32 | <ETNAn_base> + $1C10_H$ | HBG.Ethernet n*1 | — |
| HDMAC Interrupt Enable Register 1 | ETNAnHDMIER1 | 32 | <ETNAn_base> + $1C14_H$ | HBG.Ethernet n*1 | — |

Note 1. In the case of
n = 0 HBG.Ethernet0
n = 1 HBG.Ethernet1

**<ETNAn_base>**

The base address <ETNAn_base> of Ethernet is defined by the keywords in **Table 21.2, Register Base Address <ETNAn_base>**.

**Table 21.9 Register Reset Condition**

| Register Name | Reset Condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| All registers | √ | √ | √ | √ | √ |

### 21.3.2 ETNAnHDMMDRm — HDMAC Operation Mode Register m

ETNAnHDMMDRm is a 32-bit readable/writable register that specifies the operating mode of the E-DMACm.

The setting must not be changed after transmission/reception has started.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnHDMMDR0: <ETNAn_base> + $0C04_H$
ETNAnHDMMDR1: <ETNAn_base> + $1C04_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | DTOm | ITOm | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R | R |

**Table 21.10 ETNAnHDMMDRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 10 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 9 | DTOm | DMA TM OUT<br>0: If the HREADY signal which respond to DMA request is not asserted within 15 cycles, the DMA cycle is aborted by the timeout condition.<br>1: The DMA timeout is not detected and the DMA cycle is not aborted<br>**CAUTION**<br>DMA timeout function can avoid a detection of the timeout condtion only when a transfer request to GRAM from HDMAC is operated under the condition of no aribtration with other masters. When such condition is not expected on use case, set DTOm bit into 1. |
| 8 | ITOm | I/O TM OUT<br>0: If the response signal which respond to I/O access request is not asserted within 15 cycles, the I/O cycle is aborted by the timeout condition.<br>1: The I/O timeout is not detected and the I/O cycle is not aborted. |
| 7 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### 21.3.3 ETNAnHDMISRm — HDMAC Interrupt Status Register m

ETNAnHDMISRm is a 32-bit readable/writable register that shows the interrupt status of the E-DMACm.

Each bit of this register is cleared by writing 1. HDISm is cleared by clearing all bits in ETNAnEESRm register.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnHDMISR0: <ETNAn_base> + $0C10_H$
ETNAnHDMISR1: <ETNAn_base> + $1C10_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DTOSm | — | — | — | — | MSS2m | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R | R | R | R | R/W | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ITOSm | — | — | — | SLS3m | SLS2m | SLS1m | SLS0m | — | — | — | — | — | — | — | HDISm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

**Table 21.11 ETNAnHDMISRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 23 | DTOSm | DMA access Time Out<br>1: A timeout is detected during DMA access. |
| 22 to 19 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 18 | MSS2m | 1cycle Retry or Split Response<br>1: While a DMA access, the Retry or Split Response is detected only 1 cycle. The DMA behavior is the same as normal retry. |
| 17, 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 | ITOSm | I/O access Time Out<br>1: A timeout is detected during I/O access to the register. |
| 14 to 12 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 11 | SLS3m | I/O access is Seq or Busy<br>1: I/O access is Seq or Busy access. |
| 10 | SLS2m | I/O access is Burst<br>1: I/O access is burst access. |
| 9 | SLS1m | I/O access is not 32 bit<br>1: I/O access is not 32 bit. |
| 8 | SLS0m | I/O access is not 4 byte boundary<br>1: I/O access is not 4 byte boundary. |
| 7 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | HDISm | HDMAC Interrupt<br>1: HDMAC interrupt is detected. |

### 21.3.4 ETNAnHDMIERm — HDMAC Interrupt Enable Register m

ETNAnHDMIERm is a 32-bit readable/writable register that enables the interrupt output of the E-DMACm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnHDMIER0: <ETNAn_base> + $0C14_H$
ETNAnHDMIER1: <ETNAn_base> + $1C14_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DTOEm | — | — | — | — | MSE2m | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R | R | R | R | R/W | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ITOEm | — | — | — | SLE3m | SLE2m | SLE1m | SLE0m | — | — | — | — | — | — | — | HDIEm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

**Table 21.12 ETNAnHDMIERm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 23 | DTOEm | DMA access Time Out Interrupt Enable<br>1: A timeout is detected during DMA access. |
| 22 to 19 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 18 | MSE2m | 1cycle Retry or Split Response Interrupt Enable<br>1: While a DMA access, the Retry or Split Response is detected only 1 cycle. The DMA behavior is the same as normal retry. |
| 17, 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 | ITOEm | I/O access Time Out Interrupt Enable<br>1: A timeout is detected during I/O access to HDMAC or M-Port.. |
| 14 to 12 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 11 | SLE3m | I/O access is Seq or Busy Interrupt Enable<br>1: I/O access is Seq or Busy access. |
| 10 | SLE2m | I/O access is Burst Interrupt Enable<br>1: I/O access is burst access. |
| 9 | SLE1m | I/O access is not 32 bit Interrupt Enable<br>1: I/O access is not 32 bit. |
| 8 | SLE0m | I/O access is not 4 byte boundary Interrupt Enable<br>1: I/O access is not 4 byte boundary. |
| 7 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | HDIEm | HDMAC Interrupt Enable<br>1: HDMAC interrupt is enabled.<br>**Note:** This bit masks also masks any interrupt arriving from E-MAC0 (= any of events enabled in ETNAnEESIPRm like PHY link status change). |

### 21.3.5 ETNAnARSTR — Software Reset Register

ETNAnARSTR resets only TSU block in the ETNA. By writing 1 to the ARST bit in this register, a software reset is issued to TSU block of the ETNA (for 512 cycles of the HCLK). The ARST bit is always read as 0. While a software reset is issued, register access to all blocks of the ETNA is prohibited.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ETNAn_base> + $0800_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ARST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 21.13 ETNAnARSTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | ARST | Software Reset<br>When 1 is written to this bit, a software reset is issued to TSU block of the ETNA (for 512 cycles of the peripheral clock). Writing 0 does not affect this bit. This bit is always read as 0. While a software reset is issued, register access to all blocks of the ETNA is prohibited. The following registers are not initialized by a software reset.<br>ETNAnTSU_ADRH0 to ETNAnTSU_ADRH31, ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31 |

### 21.3.6 ETNAnECMRm — E-MAC Mode Register m

ETNAnECMRm is a 32-bit readable/writable register that specifies the operating mode of the MACm. The settings in this register are normally made in the initialization process following a reset.

The operating mode setting must not be changed while the transmitting and receiving functions are enabled. To switch the operating mode, return the MACm and E-DMACm to their initial states by means of the SWRTm and SWRRm bits in ETNAnEDMRm before making settings again.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnECMR0: <ETNAn_base> + $0500_H$
ETNAnECMR1: <ETNAn_base> + $1500_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TRCCMm | — | — | RCSCm | — | DPADm | RZPFm | TZPFm/DCACm | PFRm | RXFm | TXFm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | MCTm | — | — | — | MPDEm | — | — | REm | TEm | — | ILBm | — | DMm | PRMm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R | R | R | R/W | R | R | R/W | R/W | R | R/W | R | R/W | R/W |

**Table 21.14 ETNAnECMRm register contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 26 | TRCCMm | Counter Clear Mode<br>Sets the method for clearing the counter register. Refer to the description of each register.<br>0: Cleared to 0 when the relevant register is written<br>1: Cleared to 0 when the relevant register is read |
| 25, 24 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 23 | RCSCm | Checksum Calculation<br>Specifies whether to perform automatic calculation (hardware calculation) of the checksum of the receive frame data unit.<br>0: Checksum is not automatically calculated<br>1: Checksum is automatically calculated<br>Note that the checksum calculation of a frame with a VLAN tag is not supported. |
| 22 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 21 | DPADm | Data Padding<br>0: Padding is inserted to data less than 60 bytes so it is transmitted as 60-byte data<br>1: Padding is not inserted to data less than 60 bytes and it is transmitted without changes |
| 20 | RZPFm | PAUSE Frame Reception with TIME = 0<br>0: Reception of a PAUSE frame whose TIME parameter value is 0 is disabled<br>1: Reception of a PAUSE frame whose TIME parameter value is 0 is enabled |

**Table 21.14 ETNAnECMRm register contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 19 | TZPFm/DCACm | (DMm = 1)<br>PAUSE Frame Usage with TIME = 0 Enable<br>0: Control of a PAUSE frame whose TIME parameter value is 0 is disabled. The next frame is not transmitted until the time specified by the Timer value has elapsed. If a PAUSE frame whose time specified by the Timer value is 0 is received, that PAUSE frame is discarded.<br>1: Control of a PAUSE frame whose TIME parameter value is 0 is enabled. When the data size in the receive FIFO becomes smaller than the ETNAnFCFTRm setting before the time specified by the Timer value elapses, an automatic PAUSE frame with a Timer value of 0 is transmitted. On receiving a PAUSE frame with a Timer value of 0, the transmission wait state is canceled.<br><br>(DMm = 0)<br>Disable CRS Assert Check<br>1: CRS (TINT3) is not checked in half-duplex transmit mode<br>0: CRS (TINT3) is checked in half-duplex transfer mode |
| 18 | PFRm | PAUSE Frame Receive Mode<br>0: PAUSE frame is not transferred to E-DMAC<br>1: PAUSE frame is transferred to E-DMAC |
| 17 | RXFm | Operating Mode for Receiving Port Flow Control<br>0: PAUSE frame detection is disabled<br>1: Flow control for the receiving port is enabled |
| 16 | TXFm | Operating Mode for Transmitting Port Flow Control<br>0: Flow control for the transmitting port is disabled (Automatic PAUSE frame is not transmitted)<br>1: Flow control for the transmitting port is enabled (Automatic PAUSE frame is transmitted as required) |
| 15, 14 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 13 | MCTm | Multicast Address Frame Receive Mode<br>0: Frames other than the multicast address set by the CAM entry table 0 to 31 (H/L) registers are received. However, if the on-chip CAM entry table reference is disabled, all multicast address frames are received.<br>1: Only the multicast address set by the CAM entry table 0 to 31 (H/L) registers is received. |
| 12 to 10 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 9 | MPDEm | Magic Packet Detection Enable<br>Enables or disables Magic Packet detection by hardware to allow activation from the Ethernet.<br>0: Magic Packet detection is not enabled<br>1: Magic Packet detection is enabled |
| 8, 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 | REm | Reception Enable<br>If a switch is made from receiving function enabled (RE = 1) to disabled (RE = 0) while a frame is being received, the receiving function will be enabled until reception of the corresponding frame is completed.<br>0: Receiving function is disabled<br>1: Receiving function is enabled |
| 5 | TEm | Transmission Enable<br>If a switch is made from transmitting function enabled (TE0 = 1) to disabled (TE0 = 0) while a frame is being transmitted, the transmitting function will be enabled until transmission of the corresponding frame is completed.<br>0: Transmitting function is disabled<br>1: Transmitting function is enabled |
| 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

**Table 21.14 ETNAnECMRm register contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 | ILBm | Internal Loop Back Mode<br>Specifies loopback mode in the ETNA.<br>0: Normal data transmission/reception is performed<br>1: Data loopback is performed inside the MAC0 in the ETNA when DM0 = 1 |
| 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DMm | Duplex Mode<br>Specifies the ETNA transfer method.<br>0: Half-duplex transfer is specified<br>1: Full-duplex transfer is specified |
| 0 | PRMm | Promiscuous Mode<br>Setting this bit enables all Ethernet frames to be received. All Ethernet frames means all receivable frames, irrespective of differences or enabled/disabled status (destination address, broadcast address, multicast bit, etc.).<br>0: ETNA performs normal operation<br>1: ETNA performs promiscuous mode operation |

**Note:** All bits, except for TEm and REm, should be changed while the transmitting function is disabled (TEm = 0) and the receiving function is disabled (REm = 0).

## 21.3.7 ETNAnECSRm — MAC Status Register m

ETNAnECSRm is a 32-bit readable/writable register that indicates the status in the MACm. This status can be notified to the CPU by interrupts. When 1 is written to any bits of this register, the corresponding flags can be cleared. Writing 0 does not affect the flag. For bits that generate interrupts, the interrupt can be enabled or disabled by the corresponding bit in ETNAnECSIPRm.

The interrupts generated due to this status register are indicated in each ECIm bit in ETNAnEESRm of the E-DMACm for port m.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnECSR0: <ETNAn_base> + $0510_H$
ETNAnECSR1: <ETNAn_base> + $1510_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | PFROIm | — | LINKIm | MPDm | ICDm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W |

**Table 21.15 ETNAnECSRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | PFROIm | PAUSE Frame Retransmit Retry Over<br>Indicates whether the retransmit count for retransmitting a PAUSE frame when flow control is enabled has exceeded the retransmit upper-limit set in the automatic PAUSE frame retransmit count register (ETNAnTPAUSERm).<br>0: PAUSE frame retransmit count has not exceeded the upper limit<br>1: PAUSE frame retransmit count has exceeded the upper limit<br><br>For clearing this bit to 0, please refer the following procedures.<br>(1) Read TPFRCRm for clearing TPFRCR0.<br>(2) Write 1 to this bit. |
| 3 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 2 | LINKIm | ETHmLINKSTA Signal Change<br>0: ETHmLINKSTA is not changed<br>1: ETHmLINKSTA is changed |
| 1 | MPDm | Magic Packet Detection<br>Indicates that a Magic Packet has been detected on the line.<br>0: Magic Packet has not been detected<br>1: Magic Packet has been detected |
| 0 | ICDm | Illegal Carrier Detection<br>Indicates that the PHY-LSI has detected an illegal carrier on the line. If a change in the signal input from the PHY-LSI occurs in a period shorter than the software recognition period, the correct information may not be obtained. Refer to the timing specification for the PHY-LSI used.<br>0: PHY-LSI has not detected an illegal carrier on the line<br>1: PHY-LSI has detected an illegal carrier on the line |

**Note:** ETH0LINKSTA pin is not supported by P1M-C (BGA-156) and P1H-C (4MB, BGA-156).

### 21.3.8 ETNAnECSIPRm — E-MAC Interrupt Permission Register m

ETNAnECSIPRm is a 32-bit readable/writable register that enables or disables the interrupt sources indicated by ETNAnECSRm. Each bit can disable or enable interrupts corresponding to the bits in ETNAnECSRm in combination with ECImIP bit in ETNAnEESIPRm and HDIEm bit in ETNAnHDMIERm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnECSIPR0: <ETNAn_base> + $0518_H$
ETNAnECSIPR1: <ETNAn_base> + $1518_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | PFROIPm | — | LINKIPm | MPDIPm | ICDIPm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W |

**Table 21.16 ETNAnECSIPRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | PFROIPm | PAUSE Frame Retransmit Interrupt Enable<br>0: Interrupt notification by the PFROI bit is disabled<br>1: Interrupt notification by the PFROI bit is enabled |
| 3 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 2 | LINKIPm | ETHmLINKSTA Signal Change<br>0: Interrupt notification by the ETHmLINKSTA change is disabled<br>1: Interrupt notification by the ETHmLINKSTA change is enabled |
| 1 | MPDIPm | Magic Packet Detect Interrupt Enable<br>0: Interrupt notification by the MPD bit is disabled<br>1: Interrupt notification by the MPD bit is enabled |
| 0 | ICDIPm | Illegal Carrier Detect Interrupt Enable<br>0: Interrupt notification by the ICD bit is disabled<br>1: Interrupt notification by the ICD bit is enabled |

### 21.3.9 ETNAnPIRm — PHY Interface Register m

ETNAnPIRm is a 32-bit readable/writable register that provides a means of accessing the PHY-LSI internal registers via the MDIO.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnPIR0: <ETNAn_base> + $0520_H$
ETNAnPIR1: <ETNAn_base> + $1520_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MDIm | MDOm | MMDm | MDCm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.17 ETNAnPIRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 | MDIm | Management Data-In<br>Indicates the level of the ET_MDIO pin. |
| 2 | MDOm | Management Data-Out<br>Outputs the value set in this bit from the ET_MDIO pin when the MMD bit is 1. |
| 1 | MMDm | Management Mode Direction<br>Specifies the data read/write direction.<br>0: Read direction is specified<br>1: Write direction is specified |
| 0 | MDCm | Management Data Clock<br>Outputs the value set in this bit from the ET_MDC pin, and supplies the MII with the management data clock. |

## 21.3.10 ETNAnPLSRm — PHY link status Register m

ETNAnPLSRm is a 32-bit register that monitors the ETHmLINKSTA signal status.

**Access:** This register can be read in 32-bit units.

**Address:** ETNAnPLSR0: <ETNAn_base> + $0528_H$
ETNAnPLSR1: <ETNAn_base> + $1528_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | LINKm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.18 ETNAnPLSRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | LINKm | ETHmLINKSTA Status<br>For P1M-C (BGA-156) and P1H-C (4MB, BGA-156):<br>0: Channel m ETHmLINKSTA signal is “0”<br>For other products:<br>0: Channel m ETHmLINKSTA signal is “0”<br>1: Channel m ETHmLINKSTA signal is “1” |

**Note:** ETH0LINKSTA pin is not supported by P1M-C (BGA-156) and P1H-C (4MB, BGA-156).

## 21.3.11 ETNAnPIPRm — PHY interrupt polarity register m

ETNAnPIPRm is a 32-bit readable/writable register that specifies the polarity of the PHY interrupt.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnPIPR0: <ETNAn_base> + $052C_H$
ETNAnPIPR1: <ETNAn_base> + $152C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PHYILm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 21.19 ETNAnPIPRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | PHYILm | Port m PHY interrupt Polarity<br>0: Active level is Low-level<br>1: Active level is High-level |

### 21.3.12 ETNAnMAHRm — MAC Address High Register m

ETNAnMAHRm is a 32-bit readable/writable register that specifies the upper 32 bits of the 48-bit MAC address. The settings in this register are normally made in the initialization process after a reset. The MAC address setting must not be changed while the transmitting and receiving functions are enabled. Return the MACm and E-DMACm to their initial states by means of the SWRTm and SWRRm bits in ETNAnEDMRm before making settings again.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnMAHR0: <ETNAn_base> + $05C0_H$
ETNAnMAHR1: <ETNAn_base> + $15C0_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAm[47:32] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.20 ETNAnMAHRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | MAm[47:16] | MAC Address Bits 47 to 16<br>These bits are used to set the upper 32 bits of the MAC address.<br>If the MAC address is 01-23-45-67-89-AB (hexadecimal), set $01234567_H$ in this register. |

### 21.3.13 ETNAnMALRm — MAC Address Low Register m

ETNAnMALRm is a 32-bit readable/writable register that specifies the lower 16 bits of the 48-bit MAC address. The settings in this register are normally made in the initialization process after a reset. The MAC address setting must not be changed while the transmitting and receiving functions are enabled. Return the MACm and E-DMACm to their initial states by means of the SWRTm and SWRRm bits in ETNAnEDMRm before making settings again.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnMALR0: <ETNAn_base> + $05C8_H$
ETNAnMALR1: <ETNAn_base> + $15C8_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.21 ETNAnMALRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | MAm[15:0] | MAC Address Bits 15 to 0<br>These bits are used to set the lower 16 bits of the MAC address.<br>If the MAC address is 01-23-45-67-89-AB (hexadecimal), set $0000\ 89AB_H$ in this register. |

## 21.3.14 ETNAnRFLRm — Receive Frame Length Register m

ETNAnRFLRm is a 32-bit readable/writable register that specifies the maximum frame length (in bytes) that can be received. The settings in this register must not be changed while the receiving function is enabled.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnRFLR0: <ETNAn_base> + $0508_H$
ETNAnRFLR1: <ETNAn_base> + $1508_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RFLm[17:16] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFLm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.22 ETNAnRFLRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 17 to 0 | RFLm[17:0] | Receive Frame Length<br>The frame data described here refers to all fields from the destination address up to the CRC data. Frame contents from the destination address up to the data are actually transferred to memory. CRC data is not included in the transfer. When data that exceeds the specified value is received, the part of data that exceeds the specified value is discarded.<br>$00000_H$ to $005EE_H$: 1,518 bytes<br>$005EF_H$: 1,519 bytes<br>$005F0_H$: 1,520 bytes<br>: :<br>$007FF_H$: 2,047 bytes<br>$00800_H$: 2,048 bytes<br>: :<br>$01000_H$: 4,096 bytes<br>: :<br>$10000_H$: 65,536 bytes<br>: :<br>20000 to $3FFFF_H$: 131,072 bytes |

### 21.3.15 ETNAnTROCRm — Transmit Retry Over Counter Register m

ETNAnTROCRm is a 16-bit counter that indicates the number of frames that were unable to be transmitted in 16 transmission attempts including the retransfer. When 16 transmission attempts have failed, this register is incremented by 1. When the value in this register reaches 0000 $FFFF_H$, count-up is halted. This register is cleared to 0 when it is read with the TRCCMm bit in ETNAnECMRm set to 1. When the TRCCMm bit in ETNAnECMRm is 0, this register is cleared to 0 by writing any value to TROCm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTROCR0: <ETNAn_base> + $0700_H$
ETNAnTROCR1: <ETNAn_base> + $1700_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TROCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.23 ETNAnTROCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | TROCm[15:0] | Transmit Retry Over Count<br>These bits indicate the number of frames that were unable to be transmitted in 16 transmission attempts including the retransfer. |

### 21.3.16 ETNAnCDCRm — Delayed Collision Detect Counter Register m

ETNAnCDCRm is a 16-bit counter that indicates the number of all delayed collisions that occurred on the line after the start of data transmission. When the value in this register reaches 0000 $FFFF_H$, count-up is halted. This register is cleared to 0 when it is read with the TRCCMm bit in ETNAnECMRm set to 1. When the TRCCMm bit in ETNAnECMRm is 0, this register is cleared to 0 by writing any value to COSDCm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnCDCR0: <ETNAn_base> + $0708_H$
ETNAnCDCR1: <ETNAn_base> + $1708_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COSDCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.24 ETNAnCDCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | COSDCm[15:0] | Delayed Collision Detect Count<br>These bits indicate the number of all delayed collisions after the start of data transmission. |

## 21.3.17 ETNAnLCCRm — Lost Carrier Counter Register m

ETNAnLCCRm is a 16-bit counter that indicates the number of times the carrier was lost during data transmission. When the value in this register reaches 0000 $FFFF_H$, count-up is halted. This register is cleared to 0 when it is read with the TRCCMm bit in ETNAnECMRm set to 1. When the TRCCMm bit in ETNAnECMRm is 0, this register is cleared to 0 by writing any value to LCCm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnLCCR0: <ETNAn_base> + $0710_H$
ETNAnLCCR1: <ETNAn_base> + $1710_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LCCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.25 ETNAnLCCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | LCCm[15:0] | Lost Carrier Count<br>These bits indicate the number of times the carrier was lost during data transmission. |

## 21.3.18 ETNAnCEFCRm — CRC Error Frame Receive Counter Register m

ETNAnCEFCRm is a 16-bit counter that indicates the number of times a frame with a CRC error was received. When the value in this register reaches 0000 $FFFF_H$, count-up is halted. This register is cleared to 0 when it is read with the TRCCMm bit in ETNAnECMRm set to 1. When the TRCCMm bit in ETNAnECMRm is 0, this register is cleared to 0 by writing any value to CEFCm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnCEFCR0: <ETNAn_base> + $0740_H$
ETNAnCEFCR1: <ETNAn_base> + $1740_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CEFCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.26 ETNAnCEFCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | CEFCm[15:0] | CRC Error Frame Count<br>These bits indicate the number of CRC error frames received. |

### 21.3.19 ETNAnFRECRm — Frame Receive Error Counter Register m

ETNAnFRECRm is a 16-bit counter that indicates the number of frames for which a receive error was generated by the signal ETNAmRXER input from the PHY-LSI. ETNAnFRECRm is incremented each time the signal ETNAmRXER becomes active. When the value in this register reaches 0000 $FFFF_H$, count-up is halted. This register is cleared to 0 when it is read with the TRCCMm bit in ETNAnECMRm set to 1. When the TRCCMm bit in ETNAnECMRm is 0, this register is cleared to 0 by writing any value to FRECm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnFRECR0: <ETNAn_base> + $0748_H$
ETNAnFRECR1: <ETNAn_base> + $1748_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FRECm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.27 ETNAnFRECRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | FRECm[15:0] | Frame Receive Error Count<br>These bits indicate the number of errors during frame reception. |

## 21.3.20 ETNAnTSFRCRm — Too-Short Frame Receive Counter Register m

ETNAnTSFRCRm is a 16-bit counter that indicates the number of frames received with a length fewer than 64 bytes. When the value in this register reaches 0000 $FFFF_H$, count-up is halted. This register is cleared to 0 when it is read with the TRCCMm bit in ETNAnECMRm set to 1. When the TRCCMm bit in ETNAnECMRm is 0, this register is cleared to 0 by writing any value to TSFCm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSFRCR0: <ETNAn_base> + $0750_H$
ETNAnTSFRCR1: <ETNAn_base> + $1750_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSFCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.28 ETNAnTSFRCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | TSFCm[15:0] | Too-Short Frame Receive Count<br>These bits indicate the number of frames received with a length of less than 64 bytes. |

### 21.3.21 ETNAnTLFRCRm — Too-Long Frame Receive Counter Register m

ETNAnTLFRCRm is a 16-bit counter that indicates the number of frames received with a length exceeding the value specified by the receive frame length register (ETNAnRFLRm). When the value in this register reaches 0000 $FFFF_H$, count-up is halted. This register is not incremented when a frame containing residual bits is received. In this case, the reception of the frame is indicated in the residual-bit frame receive counter register (ETNAnRFCRm). This register is cleared to 0 when it is read with the TRCCMm bit in ETNAnECMRm set to 1. When the TRCCMm bit in ETNAnECMRm is 0, this register is cleared to 0 by writing any value to TLFCm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTLFRCR0: <ETNAn_base> + $0758_H$
ETNAnTLFRCR1: <ETNAn_base> + $1758_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TLFCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.29 ETNAnTLFRCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | TLFCm[15:0] | Too-Long Frame Receive Count<br>These bits indicate the number of frames received with a length exceeding the value in ETNAnRFLRm. |

## 21.3.22 ETNAnRFCRm — Residual-Bit Frame Receive Counter Register m

ETNAnRFCRm is a 16-bit counter that indicates the number of frames received containing residual bits (less than an 8-bit unit). When the value in this register reaches 0000 $FFFF_H$, count-up is halted. This register is cleared to 0 when it is read with the TRCCMm bit in ETNAnECMRm set to 1. When the TRCCMm bit in ETNAnECMRm is 0, this register is cleared to 0 by writing any value to RFCm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnRFCR0: <ETNAn_base> + $0760_H$
ETNAnRFCR1: <ETNAn_base> + $1760_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.30 ETNAnRFCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | RFCm[15:0] | Residual-Bit Frame Receive Count<br>These bits indicate the number of frames received containing residual bits. |

### 21.3.23 ETNAnMAFCRm — Multicast Address Frame Receive Counter Register m

ETNAnMAFCRm is a 16-bit counter that indicates the number of frames received with a specified multicast address. When the value in this register reaches 0000 $FFFF_H$, count-up is halted. This register is cleared to 0 when it is read with the TRCCMm bit in ETNAnECMRm set to 1. When the TRCCMm bit in ETNAnECMRm is 0, this register is cleared to 0 by writing any value to MAFCm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnMAFCR0: <ETNAn_base> + $0778_H$
ETNAnMAFCR1: <ETNAn_base> + $1778_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAFCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.31 ETNAnMAFCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | MAFCm[15:0] | Multicast Address Frame Count<br>These bits indicate the number of multicast frames received. |

### 21.3.24 ETNAnAPRm — Automatic PAUSE Frame Register m

ETNAnAPRm is used to set the TIME parameter value of an automatic PAUSE frame. When an automatic PAUSE frame is transmitted, the value set in this register is used as the TIME parameter of the PAUSE frame.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnAPR0: <ETNAn_base> + $0554_H$
ETNAnAPR1: <ETNAn_base> + $1554_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | APm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.32 ETNAnAPRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | APm[15:0] | Automatic PAUSE<br>These bits set the TIME parameter value of an automatic PAUSE frame. One bit is equivalent to 512 bit-time.<br>When flow control is enabled in transmission (PAUSE frame transmission) (TXFm bit in ETNAnECMRm = 1), set a value other than $0000_H$ in these bits.<br>$0000_H$: —<br>$0001_H$: 512 × 1 bit-time<br>$0002_H$: 512 × 2 bit-time<br>: :<br>$FFFF_H$: 512 × 65,535 bit-time<br>Note: The bit-time becomes as follows according to the transfer speed.<br>100 Mbps: 1 bit-time = 10 ns<br>10 Mbps: 1 bit-time = 100 ns |

## 21.3.25 ETNAnMPRm — Manual PAUSE Frame Register m

ETNAnMPRm is used to set the TIME parameter value of a manual PAUSE frame. When writing the Timer value to this register, manual PAUSE frame transmission is started. When a manual PAUSE frame is transmitted, the value set in this register is used as the TIME parameter of the PAUSE frame.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnMPR0: <ETNAn_base> + $0558_H$
ETNAnMPR1: <ETNAn_base> + $1558_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MPm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.33 ETNAnMPRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | MPm[15:0] | Manual PAUSE<br>These bits set the TIME parameter value of a manual PAUSE frame. One bit is equivalent to 512 bit-time.<br>$0000_H$: —<br>$0001_H$: 512 × 1 bit-time<br>$0002_H$: 512 × 2 bit-time<br>: :<br>$FFFF_H$: 512 × 65,535 bit-time<br>Note: The bit-time becomes as follows according to the transfer speed.<br>100 Mbps: 1 bit-time = 10 ns<br>10 Mbps: 1 bit-time = 100 ns |

### 21.3.26 ETNAnTPAUSERm — Automatic PAUSE Frame Retransmit Count Register m

ETNAnTPAUSERm is used to set the upper limit for the number of times to retransmit an automatic PAUSE frame. The settings in this register must not be changed while the transmitting function is enabled.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTPAUSER0: <ETNAn_base> + $0564_H$
ETNAnTPAUSER1: <ETNAn_base> + $1564_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TPAUSEm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.34 ETNAnTPAUSERm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | TPAUSEm[15:0] | Upper Limit for Automatic PAUSE Frame Retransmission<br>$0000_H$: Retransmit count is unlimited<br>$0001_H$: Retransmit count is 1<br>: :<br>$FFFF_H$: Retransmit count is 65,535 |

## 21.3.27 ETNAnTPFRCRm — Automatic PAUSE Frame Retransmit Counter Register m

ETNAnTPFRCRm is a 16-bit counter that indicates the number of times to retransmit an automatic PAUSE frame. This register is cleared to 0 when it is read or the E-HDMAC entered busy condition.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTPFRCR0: <ETNAn_base> + $0568_H$
ETNAnTPFRCR1: <ETNAn_base> + $1568_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFRTCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.35 ETNAnTPFRCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. |
| 15 to 0 | PFRTCm[15:0] | Pause Frame Retry Counter<br>The number of times to retransmit an automatic PAUSE frame between the start of a busy condition and its end. |

## 21.3.28 ETNAnPFTCRm — PAUSE Frame Transmit Counter Register m

ETNAnPFTCRm is a 16-bit counter that indicates the number of times a PAUSE frame is transmitted. This register is cleared to 0 when it is read.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnPFTCR0: <ETNAn_base> + $055C_H$
ETNAnPFTCR1: <ETNAn_base> + $155C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFTXCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.36 ETNAnPFTCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. |
| 15 to 0 | PFTXCm[15:0] | PAUSE Frame Transmit Count<br>These bits indicate the total number of automatic PAUSE frames and manual PAUSE frames transmitted. |

## 21.3.29 ETNAnPFRCRm — PAUSE Frame Receive Counter Register m

ETNAnPFRCRm is a 16-bit counter that indicates the number of times a PAUSE frame is received. This register is cleared to 0 when it is read.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnPFRCR0: <ETNAn_base> + $0560_H$
ETNAnPFRCR1: <ETNAn_base> + $1560_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFRXCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.37 ETNAnPFRCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. |
| 15 to 0 | PFRXCm[15:0] | PAUSE Frame Receive Count<br>These bits indicate the number of PAUSE frames received when flow control is enabled in reception (RXFm bit in ETNAnECMRm = 1). |

## 21.3.30 ETNAnEMRm — ETNA Mode Register m

ETNAnEMRm is used to set the operating mode of the ETNA.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnEMR0: <ETNAn_base> + $05B0_H$
ETNAnEMR1: <ETNAn_base> + $15B0_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | SPEEDm | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R |

**Table 21.38 ETNAnEMRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 2 | SPEEDm | Transfer Speed<br>SPEEDm<br>1: 100Mbps<br>0: 10Mbps |
| 1, 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### 21.3.31 ETNAnBCFRRm — Broadcast Frame Receive Count Set Register m

ETNAnBCFRRm is a register that sets the number of times broadcast frames are received consecutively.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnBCFRR0: <ETNAn_base> + $056C_H$
ETNAnBCFRR1: <ETNAn_base> + $156C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BCFm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.39 ETNAnBCFRRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | BCFm[15:0] | Consecutive Broadcast Frame Reception Count Setting<br>ETNA can receive frames with the broadcast address consecutively until the number of receptions reaches the value set in these bits. If the number of consecutive broadcast frame reception exceeds BCFm configuration, the subsequent broadcast frames are discarded.<br>$0000_H$: No limitation on the number of receptions<br>$0001_H$: Receive one frame<br>: :<br>$FFFF_H$: Receive 65535 frames |

## 21.3.32 ETNAnCRBCFRRm — Count Reset of BCFRR m

ETNAnCRBCFRRm counts the number of times that consecutive broadcast frame reception exceeds the times defined by ETNAnBCFRRm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnCRBCFRR0: <ETNAn_base> + $0570_H$
ETNAnCRBCFRR1: <ETNAn_base> + $1570_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNTm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.40 ETNAnCRBCFRRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | CNTm[15:0] | Reset number counter of ETNAnBCFRRm<br>CNTm is incremented only when the number of times that consecutive broadcast frame reception exceeds ETNAnBCFRRm configuration. If CNTm become $FFFF_H$, count-up is halted.<br>In case the final reception frame is the broadcast frame, CNTm is not incremented until non broadcast frame is received. |

### 21.3.33 ETNAnTSU_CTRST — TSU Counter Reset Register

ETNAnTSU_CTRST clears the transmit and receive frame counters to 0.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSU_CTRST: <ETNAn_base> + $0804_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CTRST | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R |

**Table 21.41 ETNAnTSU_CTRST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 8 | CTRST | TSU Counter Reset<br>When 1 is written to this bit, the values of registers ETNAnTXNLCR0/ ETNAnTXNLCR1, ETNAnTXALCR0/ETNAnTXALCR1, ETNAnRXNLCR0/ ETNAnRXNLCR1, ETNAnRXALCR0/ETNAnRXALCR1 are cleared to 0. Writing 0 does not affect this bit. This bit is always read as 0. |
| 7 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

## 21.3.34 ETNAnTSU_FWSLm — Receive Function Set Register

ETNAnTSU_FWSLm sets the processing method of each frame in port m reception.
This register setting must not be changed while the transmitting and receiving functions are enabled. Return the MACm and E-DMACm to their initial states by means of the SWRTm and SWRRm bits in ETNAnEDMRm before making settings again.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSU_FWSL0: <ETNAn_base> + $0830_H$
ETNAnTSU_FWSL1: <ETNAn_base> + $0834_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | RMSAm | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R |

**Table 21.42 ETNAnTSU_FWSL0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | RMSAm | Receive Mode using Source Address<br>0: SA (source address) has no effect for the frame reception.<br>1: SA (source address) has effect for the frame reception as follows. Frames that are discarded by VLAN-filtering are not received. |
| 3 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

| SA (source address) is existed in the entry table | DA(destination address) is existed in the entry | Received |
|---|---|---|
| | DA(destination address) is not existed in the entry | Discarded |
| SA (source address) is not existed in the entry table | | Received |

### 21.3.35 ETNAnTSU_FWSLC — TSU Function Set Register (Common)

When the CAM is used, the referred area in the CAM entry table (partially or wholly) can be specified by the ETNAnTSU_POST1 to ETNAnTSU_POST4 registers. ETNAnTSU_FWSLC enables settings by the ETNAnTSU_POST1 to ETNAnTSU_POST4 registers. This register setting must not be changed while the transmitting and receiving functions are enabled. Return the MACm and E-DMACm to their initial states by means of the SWRTm and SWRRm bits in ETNAnEDMRm before making settings again.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSU_FWSLC: <ETNAn_base> + $0838_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | POSTENU | POSTENL | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.43 ETNAnTSU_FWSLC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 13 | POSTENU | Enables the settings of the POST field of CAM entry tables 0 to 15 (settings by the ETNAnTSU_POST1 and ETNAnTSU_POST2 registers).<br>0: Disables the settings of the POST field. (The CAM entry table is referred to only in port 0 reception.)<br>1: Enables the settings of the POST field. (The CAM entry table reference conditions follow the POST field settings.) |
| 12 | POSTENL | Enables the settings of the POST field of CAM entry tables 16 to 31 (settings by the ETNAnTSU_POST3 and ETNAnTSU_POST4 registers).<br>0: Disables the settings of the POST field. (The CAM entry table is referred to only in port 1 reception.)<br>1: Enables the settings of the POST field. (The CAM entry table reference conditions follow the POST field settings.) |
| 11 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

## 21.3.36 ETNAnTSU_FWSR — TSU Status Register

ETNAnTSU_FWSR is a 32-bit readable/writable register that indicates the status during TSU operations.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSU_FWSR: <ETNAn_base> + $0850_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | RINT50 | RINT40 | RINT30 | RINT20 | RINT10 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | RINT51 | RINT41 | RINT31 | RINT21 | RINT11 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 21.44 ETNAnTSU_FWSR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 20 | RINT50 | MAC0 Residual-Bit Frame Receive<br>Set to 1 when a frame containing residual bits (less than an 8-bit unit) is received in the MAC0. |
| 19 | RINT40 | MAC0 Too-Long Frame Receive<br>Set to 1 when a frame exceeding the value set by ETNAnRFLR0 is received in the MAC0. |
| 18 | RINT30 | MAC0 Too-Short Frame Receive<br>Set to 1 when a frame with a length of less than 64 bytes is received in the MAC0. |
| 17 | RINT20 | MAC0 Frame Receive Error<br>Set to 1 when a receive error is detected on the signal RX_ER from the PHY-LSI (ETNAmRXER) in the MAC0. |
| 16 | RINT10 | MAC0 CRC Error Frame Receive<br>Set to 1 when a receive frame results in a CRC error in the MAC0. |
| 15 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | RINT51 | MAC1 Residual-Bit Frame Receive<br>Set to 1 when a frame containing residual bits (less than an 8-bit unit) is received in the MAC1. |
| 3 | RINT41 | MAC1 Too-Long Frame Receive<br>Set to 1 when a frame exceeding the value set by ETNAnRFLR1 is received in the MAC1. |
| 2 | RINT31 | MAC1 Too-Short Frame Receive<br>Set to 1 when a frame with a length of less than 64 bytes is received in the MAC1. |
| 1 | RINT21 | MAC1 Frame Receive Error<br>Set to 1 when a receive error is detected on the signal RX_ER from the PHY-LSI (ETNAmRXER) in the MAC1. |
| 0 | RINT11 | MAC1 CRC Error Frame Receive<br>Set to 1 when a receive frame results in a CRC error in the MAC1. |

## 21.3.37 ETNAnTSU_FWINMK — TSU Status Interrupt Mask Register

ETNAnTSU_FWINMK is a 32-bit readable/writable register that sets the interrupt mask for status bits in ETNAnTSU_FWSR.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSU_FWINMK: <ETNAn_base> + $0854_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | RINTM50 | RINTM40 | RINTM30 | RINTM20 | RINTM10 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | RINTM51 | RINTM41 | RINTM31 | RINTM21 | RINTM11 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 21.45 ETNAnTSU_FWINMK register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 20 | RINTM50 | MAC0 Residual-Bit Frame Receive Interrupt Mask<br>0: Interrupts disabled<br>1: Interrupts enabled |
| 19 | RINTM40 | MAC0 Too-Long Frame Receive Interrupt Mask<br>0: Interrupts disabled<br>1: Interrupts enabled |
| 18 | RINTM30 | MAC0 Too-Short Frame Receive Interrupt Mask<br>0: Interrupts disabled<br>1: Interrupts enabled |
| 17 | RINTM20 | MAC0 Frame Receive Error Interrupt Mask<br>0: Interrupts disabled<br>1: Interrupts enabled |
| 16 | RINTM10 | MAC0 CRC Error Frame Receive Interrupt Mask<br>0: Interrupts disabled<br>1: Interrupts enabled |
| 15 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | RINTM51 | MAC1 Residual-Bit Frame Receive Interrupt Mask<br>0: Interrupts disabled<br>1: Interrupts enabled |
| 3 | RINTM41 | MAC1 Too-Long Frame Receive Interrupt Mask<br>0: Interrupts disabled<br>1: Interrupts enabled |
| 2 | RINTM31 | MAC1 Too-Short Frame Receive Interrupt Mask<br>0: Interrupts disabled<br>1: Interrupts enabled |
| 1 | RINTM21 | MAC1 Frame Receive Error Interrupt Mask<br>0: Interrupts disabled<br>1: Interrupts enabled |

**Table 21.45 ETNAnTSU_FWINMK register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | RINTM11 | MAC1 CRC Error Frame Receive Interrupt Mask<br>0: Interrupts disabled<br>1: Interrupts enabled |

## 21.3.38 ETNAnTSU_VTAGm — VLANtag Set Register

ETNAnTSU_VTAGm enables or disables the frame receive/discard evaluation function based on the VLAN number in port m receive operations, and also sets the VLAN number. This register setting must not be changed while the transmitting and receiving functions are enabled. Return the MACm and E-DMACm to their initial states by means of the SWRTm and SWRRm bits in ETNAnEDMRm before making settings again.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSU_VTAG0: <ETNAn_base> + $0858_H$
ETNAnTSU_VTAG1: <ETNAn_base> + $085C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VTAGm | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VIDm | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.46 ETNAnTSU_VTAGm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | VTAGm | Port m VLAN tag Evaluation Function<br>0: Disables receive/discard evaluation for frames based on the VLAN number<br>1: Enables receive/discard evaluation for frames based on the VLAN number |
| 30 to 12 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 11 to 0 | VIDm | V-LAN ID Setting (VID)<br>These bits set the VLAN number received by port m receive frames. |

## 21.3.39 ETNAnTSU_ADSBSY — CAM Entry Table Busy Register

When CAM entry table registers (ETNAnTSU_ADRH0 to ETNAnTSU_ADRH31 and ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31) are set by register writing, the ADSBSY bit in this register is set to 1 (when the process of reflecting the contents of the CAM entry table registers in the CAM controller is completed inside the TSU, the ADSBSY bit is automatically restored to 0).

Access to ETNAnTSU_ADRH0 to ETNAnTSU_ADRH31 and ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31 is prohibited, while the ADSBSY bit in this register is set to 1. This register is a read-only status register, which must not be written to.

**Access:** This register can be read in 32-bit units.

**Address:** ETNAnTSU_ADSBSY: <ETNAn_base> + $0860_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ADS BSY |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.47 ETNAnTSU_ADSBSY register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ADSBSY | CAM Entry Table Setting Busy<br>When ETNAnTSU_ADRH0 to ETNAnTSU_ADRH31 and ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31 are set by register writing during the MAC0/1 are receiving any frame, this bit is set to 1. When the process of reflecting the contents of the CAM entry table registers in the CAM controller is completed inside the TSU, this bit is automatically restored to 0. Access to ETNAnTSU_ADRH0 to ETNAnTSU_ADRH31 and ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31 is prohibited, while this bit is set to 1. Writing to this register is also prohibited. |

## 21.3.40 ETNAnTSU_TEN — CAM Entry Table Enable Register

ETNAnTSU_TEN enables or disables the settings of ETNAnTSU_ADRH0 to ETNAnTSU_ADRH31 and ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSU_TEN: <ETNAn_base> + $0864_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEN0 | TEN1 | TEN2 | TEN3 | TEN4 | TEN5 | TEN6 | TEN7 | TEN8 | TEN9 | TEN10 | TEN11 | TEN12 | TEN13 | TEN14 | TEN15 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEN16 | TEN17 | TEN18 | TEN19 | TEN20 | TEN21 | TEN22 | TEN23 | TEN24 | TEN25 | TEN26 | TEN27 | TEN28 | TEN29 | TEN30 | TEN31 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.48 ETNAnTSU_TEN register contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | TEN0 | CAM Entry Table 0 (ETNAnTSU_ADRH0 and ETNAnTSU_ADRL0) Setting<br>0: Disabled<br>1: Enabled |
| 30 | TEN1 | CAM Entry Table 1 (ETNAnTSU_ADRH1 and ETNAnTSU_ADRL1) Setting<br>0: Disabled<br>1: Enabled |
| 29 | TEN2 | CAM Entry Table 2 (ETNAnTSU_ADRH2 and ETNAnTSU_ADRL2) Setting<br>0: Disabled<br>1: Enabled |
| 28 | TEN3 | CAM Entry Table 3 (ETNAnTSU_ADRH3 and ETNAnTSU_ADRL3) Setting<br>0: Disabled<br>1: Enabled |
| 27 | TEN4 | CAM Entry Table 4 (ETNAnTSU_ADRH4 and ETNAnTSU_ADRL4) Setting<br>0: Disabled<br>1: Enabled |
| 26 | TEN5 | CAM Entry Table 5 (ETNAnTSU_ADRH5 and ETNAnTSU_ADRL5) Setting<br>0: Disabled<br>1: Enabled |
| 25 | TEN6 | CAM Entry Table 6 (ETNAnTSU_ADRH6 and ETNAnTSU_ADRL6) Setting<br>0: Disabled<br>1: Enabled |
| 24 | TEN7 | CAM Entry Table 7 (ETNAnTSU_ADRH7 and ETNAnTSU_ADRL7) Setting<br>0: Disabled<br>1: Enabled |
| 23 | TEN8 | CAM Entry Table 8 (ETNAnTSU_ADRH8 and ETNAnTSU_ADRL8) Setting<br>0: Disabled<br>1: Enabled |
| 22 | TEN9 | CAM Entry Table 9 (ETNAnTSU_ADRH9 and ETNAnTSU_ADRL9) Setting<br>0: Disabled<br>1: Enabled |
| 21 | TEN10 | CAM Entry Table 10 (ETNAnTSU_ADRH10 and ETNAnTSU_ADRL10) Setting<br>0: Disabled<br>1: Enabled |

Table 21.48 ETNAnTSU_TEN register contents (2/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 20 | TEN11 | CAM Entry Table 11 (ETNAnTSU_ADRH11 and ETNAnTSU_ADRL11) Setting<br>0: Disabled<br>1: Enabled |
| 19 | TEN12 | CAM Entry Table 12 (ETNAnTSU_ADRH12 and ETNAnTSU_ADRL12) Setting<br>0: Disabled<br>1: Enabled |
| 18 | TEN13 | CAM Entry Table 13 (ETNAnTSU_ADRH13 and ETNAnTSU_ADRL13) Setting<br>0: Disabled<br>1: Enabled |
| 17 | TEN14 | CAM Entry Table 14 (ETNAnTSU_ADRH14 and ETNAnTSU_ADRL14) Setting<br>0: Disabled<br>1: Enabled |
| 16 | TEN15 | CAM Entry Table 15 (ETNAnTSU_ADRH15 and ETNAnTSU_ADRL15) Setting<br>0: Disabled<br>1: Enabled |
| 15 | TEN16 | CAM Entry Table 16 (ETNAnTSU_ADRH16 and ETNAnTSU_ADRL16) Setting<br>0: Disabled<br>1: Enabled |
| 14 | TEN17 | CAM Entry Table 17 (ETNAnTSU_ADRH17 and ETNAnTSU_ADRL17) Setting<br>0: Disabled<br>1: Enabled |
| 13 | TEN18 | CAM Entry Table 18 (ETNAnTSU_ADRH18 and ETNAnTSU_ADRL18) Setting<br>0: Disabled<br>1: Enabled |
| 12 | TEN19 | CAM Entry Table 19 (ETNAnTSU_ADRH19 and ETNAnTSU_ADRL19) Setting<br>0: Disabled<br>1: Enabled |
| 11 | TEN20 | CAM Entry Table 20 (ETNAnTSU_ADRH20 and ETNAnTSU_ADRL20) Setting<br>0: Disabled<br>1: Enabled |
| 10 | TEN21 | CAM Entry Table 21 (ETNAnTSU_ADRH21 and ETNAnTSU_ADRL21) Setting<br>0: Disabled<br>1: Enabled |
| 9 | TEN22 | CAM Entry Table 22 (ETNAnTSU_ADRH22 and ETNAnTSU_ADRL22) Setting<br>0: Disabled<br>1: Enabled |
| 8 | TEN23 | CAM Entry Table 23 (ETNAnTSU_ADRH23 and ETNAnTSU_ADRL23) Setting<br>0: Disabled<br>1: Enabled |
| 7 | TEN24 | CAM Entry Table 24 (ETNAnTSU_ADRH24 and ETNAnTSU_ADRL24) Setting<br>0: Disabled<br>1: Enabled |
| 6 | TEN25 | CAM Entry Table 25 (ETNAnTSU_ADRH25 and ETNAnTSU_ADRL25) Setting<br>0: Disabled<br>1: Enabled |
| 5 | TEN26 | CAM Entry Table 26 (ETNAnTSU_ADRH26 and ETNAnTSU_ADRL26) Setting<br>0: Disabled<br>1: Enabled |
| 4 | TEN27 | CAM Entry Table 27 (ETNAnTSU_ADRH27 and ETNAnTSU_ADRL27) Setting<br>0: Disabled<br>1: Enabled |
| 3 | TEN28 | CAM Entry Table 28 (ETNAnTSU_ADRH28 and ETNAnTSU_ADRL28) Setting<br>0: Disabled<br>1: Enabled |
| 2 | TEN29 | CAM Entry Table 29 (ETNAnTSU_ADRH29 and ETNAnTSU_ADRL29) Setting<br>0: Disabled<br>1: Enabled |

**Table 21.48 ETNAnTSU_TEN register contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | TEN30 | CAM Entry Table 30 (ETNAnTSU_ADRH30 and ETNAnTSU_ADRL30) Setting<br>0: Disabled<br>1: Enabled |
| 0 | TEN31 | CAM Entry Table 31 (ETNAnTSU_ADRH31 and ETNAnTSU_ADRL31) Setting<br>0: Disabled<br>1: Enabled |

## 21.3.41 ETNAnTSU_POST1 — CAM Entry Table POST1 Register

When using the CAM, the conditions for referring to each CAM entry table can be specified by using the ETNAnTSU_POST1 to ETNAnTSU_POST4 registers. ETNAnTSU_POST1 specifies the conditions for referring to ETNAnTSU_ADRH0 to ETNAnTSU_ADRH7 and ETNAnTSU_ADRL0 to ETNAnTSU_ADRL7. The settings of this register are valid when the POSTENU bit in ETNAnTSU_FWSLC is set to 1.

This register setting must not be changed while the transmitting and receiving functions are enabled. Return the MACm and E-DMACm to their initial states by means of the SWRTm and SWRRm bits in ETNAnEDMRm before making settings again.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSU_POST1: <ETNAn_base> + $0870_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | POST0[3:0] | | | | POST1[3:0] | | | | POST2[3:0] | | | | POST3[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | POST4[3:0] | | | | POST5[3:0] | | | | POST6[3:0] | | | | POST7[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R |

**Table 21.49 ETNAnTSU_POST1 register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | POST0[3:0] | These bits set the conditions for referring to CAM entry table 0. By setting multiple bits to 1, multiple conditions can be selected.<br>POST0[3]: CAM entry table 0 is referred to in port 0 reception.<br>POST0[1]: CAM entry table 0 is referred to in port 1 reception. |
| 27 to 24 | POST1[3:0] | These bits set the conditions for referring to CAM entry table 1. By setting multiple bits to 1, multiple conditions can be selected.<br>POST1[3]: CAM entry table 1 is referred to in port 0 reception.<br>POST1[1]: CAM entry table 1 is referred to in port 1 reception. |
| 23 to 20 | POST2[3:0] | These bits set the conditions for referring to CAM entry table 2. By setting multiple bits to 1, multiple conditions can be selected.<br>POST2[3]: CAM entry table 2 is referred to in port 0 reception.<br>POST2[1]: CAM entry table 2 is referred to in port 1 reception. |
| 19 to 16 | POST3[3:0] | These bits set the conditions for referring to CAM entry table 3. By setting multiple bits to 1, multiple conditions can be selected.<br>POST3[3]: CAM entry table 3 is referred to in port 0 reception.<br>POST3[1]: CAM entry table 3 is referred to in port 1 reception. |
| 15 to 12 | POST4[3:0] | These bits set the conditions for referring to CAM entry table 4. By setting multiple bits to 1, multiple conditions can be selected.<br>POST4[3]: CAM entry table 4 is referred to in port 0 reception.<br>POST4[1]: CAM entry table 4 is referred to in port 1 reception. |
| 11 to 8 | POST5[3:0] | These bits set the conditions for referring to CAM entry table 5. By setting multiple bits to 1, multiple conditions can be selected.<br>POST5[3]: CAM entry table 5 is referred to in port 0 reception.<br>POST5[1]: CAM entry table 5 is referred to in port 1 reception. |

**Table 21.49 ETNAnTSU_POST1 register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | POST6[3:0] | These bits set the conditions for referring to CAM entry table 6. By setting multiple bits to 1, multiple conditions can be selected.<br>POST6[3]: CAM entry table 6 is referred to in port 0 reception.<br>POST6[1]: CAM entry table 6 is referred to in port 1 reception. |
| 3 to 0 | POST7[3:0] | These bits set the conditions for referring to CAM entry table 7. By setting multiple bits to 1, multiple conditions can be selected.<br>POST7[3]: CAM entry table 7 is referred to in port 0 reception.<br>POST7[1]: CAM entry table 7 is referred to in port 1 reception. |

## 21.3.42 ETNAnTSU_POST2 — CAM Entry Table POST2 Register

When using the CAM, the conditions for referring to each CAM entry table can be specified by using the ETNAnTSU_POST1 to ETNAnTSU_POST4 registers. ETNAnTSU_POST2 specifies the conditions for referring to ETNAnTSU_ADRH8 to ETNAnTSU_ADRH15 and ETNAnTSU_ADRL8 to ETNAnTSU_ADRL15. The settings of this register are valid when the POSTENU bit in ETNAnTSU_FWSLC is set to 1.

This register setting must not be changed while the transmitting and receiving functions are enabled. Return the MACm and E-DMACm to their initial states by means of the SWRTm and SWRRm bits in ETNAnEDMRm before making settings again.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSU_POST2: <ETNAn_base> + $0874_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | POST8[3:0] | | | | POST9[3:0] | | | | POST10[3:0] | | | | POST11[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | POST12[3:0] | | | | POST13[3:0] | | | | POST14[3:0] | | | | POST15[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R |

**Table 21.50 ETNAnTSU_POST2 register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | POST8[3:0] | These bits set the conditions for referring to CAM entry table 8. By setting multiple bits to 1, multiple conditions can be selected.<br>POST8[3]: CAM entry table 8 is referred to in port 0 reception.<br>POST8[1]: CAM entry table 8 is referred to in port 1 reception. |
| 27 to 24 | POST9[3:0] | These bits set the conditions for referring to CAM entry table 9. By setting multiple bits to 1, multiple conditions can be selected.<br>POST9[3]: CAM entry table 9 is referred to in port 0 reception.<br>POST9[1]: CAM entry table 9 is referred to in port 1 reception. |
| 23 to 20 | POST10[3:0] | These bits set the conditions for referring to CAM entry table 10. By setting multiple bits to 1, multiple conditions can be selected.<br>POST10[3]: CAM entry table 10 is referred to in port 0 reception.<br>POST10[1]: CAM entry table 10 is referred to in port 1 reception. |
| 19 to 16 | POST11[3:0] | These bits set the conditions for referring to CAM entry table 11. By setting multiple bits to 1, multiple conditions can be selected.<br>POST11[3]: CAM entry table 11 is referred to in port 0 reception.<br>POST11[1]: CAM entry table 11 is referred to in port 1 reception. |
| 15 to 12 | POST12[3:0] | These bits set the conditions for referring to CAM entry table 12. By setting multiple bits to 1, multiple conditions can be selected.<br>POST12[3]: CAM entry table 12 is referred to in port 0 reception.<br>POST12[1]: CAM entry table 12 is referred to in port 1 reception. |
| 11 to 8 | POST13[3:0] | These bits set the conditions for referring to CAM entry table 13. By setting multiple bits to 1, multiple conditions can be selected.<br>POST13[3]: CAM entry table 13 is referred to in port 0 reception.<br>POST13[1]: CAM entry table 13 is referred to in port 1 reception. |

**Table 21.50 ETNAnTSU_POST2 register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | POST14[3:0] | These bits set the conditions for referring to CAM entry table 14. By setting multiple bits to 1, multiple conditions can be selected.<br>POST14[3]: CAM entry table 14 is referred to in port 0 reception.<br>POST14[1]: CAM entry table 14 is referred to in port 1 reception. |
| 3 to 0 | POST15[3:0] | These bits set the conditions for referring to CAM entry table 15. By setting multiple bits to 1, multiple conditions can be selected.<br>POST15[3]: CAM entry table 15 is referred to in port 0 reception.<br>POST15[1]: CAM entry table 15 is referred to in port 1 reception. |

## 21.3.43 ETNAnTSU_POST3 — CAM Entry Table POST3 Register

When using the CAM, the conditions for referring to each CAM entry table can be specified by using the ETNAnTSU_POST1 to ETNAnTSU_POST4 registers. ETNAnTSU_POST3 specifies the conditions for referring to ETNAnTSU_ADRH16 to ETNAnTSU_ADRH23 and ETNAnTSU_ADRL16 to ETNAnTSU_ADRL23. The settings of this register are valid when the POSTENL bit in ETNAnTSU_FWSLC is set to 1.

This register setting must not be changed while the transmitting and receiving functions are enabled. Return the MACm and E-DMACm to their initial states by means of the SWRTm and SWRRm bits in ETNAnEDMRm before making settings again.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSU_POST3: <ETNAn_base> + $0878_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | POST16[3:0] | | | | POST17[3:0] | | | | POST18[3:0] | | | | POST19[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | POST20[3:0] | | | | POST21[3:0] | | | | POST22[3:0] | | | | POST23[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R |

**Table 21.51 ETNAnTSU_POST3 register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | POST16[3:0] | These bits set the conditions for referring to CAM entry table 16. By setting multiple bits to 1, multiple conditions can be selected.<br>POST16[3]: CAM entry table 16 is referred to in port 0 reception.<br>POST16[1]: CAM entry table 16 is referred to in port 1 reception. |
| 27 to 24 | POST17[3:0] | These bits set the conditions for referring to CAM entry table 17. By setting multiple bits to 1, multiple conditions can be selected.<br>POST17[3]: CAM entry table 17 is referred to in port 0 reception.<br>POST17[1]: CAM entry table 17 is referred to in port 1 reception. |
| 23 to 20 | POST18[3:0] | These bits set the conditions for referring to CAM entry table 18. By setting multiple bits to 1, multiple conditions can be selected.<br>POST18[3]: CAM entry table 18 is referred to in port 0 reception.<br>POST18[1]: CAM entry table 18 is referred to in port 1 reception. |
| 19 to 16 | POST19[3:0] | These bits set the conditions for referring to CAM entry table 19. By setting multiple bits to 1, multiple conditions can be selected.<br>POST19[3]: CAM entry table 19 is referred to in port 0 reception.<br>POST19[1]: CAM entry table 19 is referred to in port 1 reception. |
| 15 to 12 | POST20[3:0] | These bits set the conditions for referring to CAM entry table 20. By setting multiple bits to 1, multiple conditions can be selected.<br>POST20[3]: CAM entry table 20 is referred to in port 0 reception.<br>POST20[1]: CAM entry table 20 is referred to in port 1 reception. |
| 11 to 8 | POST21[3:0] | These bits set the conditions for referring to CAM entry table 21. By setting multiple bits to 1, multiple conditions can be selected.<br>POST21[3]: CAM entry table 21 is referred to in port 0 reception.<br>POST21[1]: CAM entry table 21 is referred to in port 1 reception. |

**Table 21.51 ETNAnTSU_POST3 register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | POST22[3:0] | These bits set the conditions for referring to CAM entry table 22. By setting multiple bits to 1, multiple conditions can be selected.<br>POST22[3]: CAM entry table 22 is referred to in port 0 reception.<br>POST22[1]: CAM entry table 22 is referred to in port 1 reception. |
| 3 to 0 | POST23[3:0] | These bits set the conditions for referring to CAM entry table 23. By setting multiple bits to 1, multiple conditions can be selected.<br>POST23[3]: CAM entry table 23 is referred to in port 0 reception.<br>POST23[1]: CAM entry table 23 is referred to in port 1 reception. |

## 21.3.44 ETNAnTSU_POST4 — CAM Entry Table POST4 Register

When using the CAM, the conditions for referring to each CAM entry table can be specified by using the ETNAnTSU_POST1 to ETNAnTSU_POST4 registers. ETNAnTSU_POST4 specifies the conditions for referring to ETNAnTSU_ADRH24 to ETNAnTSU_ADRH31 and ETNAnTSU_ADRL24 to ETNAnTSU_ADRL31. The settings of this register are valid when the POSTENL bit in ETNAnTSU_FWSLC is set to 1.

This register setting must not be changed while the transmitting and receiving functions are enabled. Return the MACm and E-DMACm to their initial states by means of the SWRTm and SWRRm bits in ETNAnEDMRm before making settings again.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTSU_POST4: <ETNAn_base> + $087C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | POST24[3:0] | | | | POST25[3:0] | | | | POST26[3:0] | | | | POST27[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | POST28[3:0] | | | | POST29[3:0] | | | | POST30[3:0] | | | | POST31[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R | R/W | R |

**Table 21.52 ETNAnTSU_POST4 register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | POST24[3:0] | These bits set the conditions for referring to CAM entry table 24. By setting multiple bits to 1, multiple conditions can be selected.<br>POST24[3]: CAM entry table 24 is referred to in port 0 reception.<br>POST24[1]: CAM entry table 24 is referred to in port 1 reception. |
| 27 to 24 | POST25[3:0] | These bits set the conditions for referring to CAM entry table 25. By setting multiple bits to 1, multiple conditions can be selected.<br>POST25[3]: CAM entry table 25 is referred to in port 0 reception.<br>POST25[1]: CAM entry table 25 is referred to in port 1 reception. |
| 23 to 20 | POST26[3:0] | These bits set the conditions for referring to CAM entry table 26. By setting multiple bits to 1, multiple conditions can be selected.<br>POST26[3]: CAM entry table 26 is referred to in port 0 reception.<br>POST26[1]: CAM entry table 26 is referred to in port 1 reception. |
| 19 to 16 | POST27[3:0] | These bits set the conditions for referring to CAM entry table 27. By setting multiple bits to 1, multiple conditions can be selected.<br>POST27[3]: CAM entry table 27 is referred to in port 0 reception.<br>POST27[1]: CAM entry table 27 is referred to in port 1 reception. |
| 15 to 12 | POST28[3:0] | These bits set the conditions for referring to CAM entry table 28. By setting multiple bits to 1, multiple conditions can be selected.<br>POST28[3]: CAM entry table 28 is referred to in port 0 reception.<br>POST28[1]: CAM entry table 28 is referred to in port 1 reception. |
| 11 to 8 | POST29[3:0] | These bits set the conditions for referring to CAM entry table 29. By setting multiple bits to 1, multiple conditions can be selected.<br>POST29[3]: CAM entry table 29 is referred to in port 0 reception.<br>POST29[1]: CAM entry table 29 is referred to in port 1 reception. |

**Table 21.52 ETNAnTSU_POST4 register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | POST30[3:0] | These bits set the conditions for referring to CAM entry table 30. By setting multiple bits to 1, multiple conditions can be selected.<br>POST30[3]: CAM entry table 30 is referred to in port 0 reception.<br>POST30[1]: CAM entry table 30 is referred to in port 1 reception. |
| 3 to 0 | POST31[3:0] | These bits set the conditions for referring to CAM entry table 31. By setting multiple bits to 1, multiple conditions can be selected.<br>POST31[3]: CAM entry table 31 is referred to in port 0 reception.<br>POST31[1]: CAM entry table 31 is referred to in port 1 reception. |

## 21.3.45 ETNAnTSU_ADRH0 to ETNAnTSU_ADRH31 — CAM Entry Table 0H to 31H Registers

ETNAnTSU_ADRH0 to ETNAnTSU_ADRH31 are entry tables referred to by the CAM in reception. Each of these registers sets the upper 32 bits of the 48-bit MAC address. Maximum 32 entries of MAC addresses can be registered. The CAM entry rable setting must not be changed while the transmitting and receiving functions are enabled. ETNAnTSU_ADRHm is written at the same time with ETNAnTSU_ADRLm writing which is given after ETNAnTSU_ADRHm writing.

**Access:** This register can be read/written in 32-bit units.

**Address:**

| | |
|---|---|
| ETNAnTSU_ADRH0: <ETNAn_base> + $0900_H$ | ETNAnTSU_ADRH1: <ETNAn_base> + $0908_H$ |
| ETNAnTSU_ADRH2: <ETNAn_base> + $0910_H$ | ETNAnTSU_ADRH3: <ETNAn_base> + $0918_H$ |
| ETNAnTSU_ADRH4: <ETNAn_base> + $0920_H$ | ETNAnTSU_ADRH5: <ETNAn_base> + $0928_H$ |
| ETNAnTSU_ADRH6: <ETNAn_base> + $0930_H$ | ETNAnTSU_ADRH7: <ETNAn_base> + $0938_H$ |
| ETNAnTSU_ADRH8: <ETNAn_base> + $0940_H$ | ETNAnTSU_ADRH9: <ETNAn_base> + $0948_H$ |
| ETNAnTSU_ADRH10: <ETNAn_base> + $0950_H$ | ETNAnTSU_ADRH11: <ETNAn_base> + $0958_H$ |
| ETNAnTSU_ADRH12: <ETNAn_base> + $0960_H$ | ETNAnTSU_ADRH13: <ETNAn_base> + $0968_H$ |
| ETNAnTSU_ADRH14: <ETNAn_base> + $0970_H$ | ETNAnTSU_ADRH15: <ETNAn_base> + $0978_H$ |
| ETNAnTSU_ADRH16: <ETNAn_base> + $0980_H$ | ETNAnTSU_ADRH17: <ETNAn_base> + $0988_H$ |
| ETNAnTSU_ADRH18: <ETNAn_base> + $0990_H$ | ETNAnTSU_ADRH19: <ETNAn_base> + $0998_H$ |
| ETNAnTSU_ADRH20: <ETNAn_base> + $09A0_H$ | ETNAnTSU_ADRH21: <ETNAn_base> + $09A8_H$ |
| ETNAnTSU_ADRH22: <ETNAn_base> + $09B0_H$ | ETNAnTSU_ADRH23: <ETNAn_base> + $09B8_H$ |
| ETNAnTSU_ADRH24: <ETNAn_base> + $09C0_H$ | ETNAnTSU_ADRH25: <ETNAn_base> + $09C8_H$ |
| ETNAnTSU_ADRH26: <ETNAn_base> + $09D0_H$ | ETNAnTSU_ADRH27: <ETNAn_base> + $09D8_H$ |
| ETNAnTSU_ADRH28: <ETNAn_base> + $09E0_H$ | ETNAnTSU_ADRH29: <ETNAn_base> + $09E8_H$ |
| ETNAnTSU_ADRH30: <ETNAn_base> + $09F0_H$ | ETNAnTSU_ADRH31: <ETNAn_base> + $09F8_H$ |

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADRHi[31:16] (i = 0 to 31) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADRH i[15:0] (i = 0 to 31) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.53 ETNAnTSU_ADRHm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ADRH i [31:0]<br>(i: 0 to 31) | MAC Address Bits<br>These bits set the upper 32 bits of the MAC address.<br>When the MAC address is 01-23-45-67-89-AB (displayed in hexadecimal), set $01234567_H$ in this register. |

**Note:** Set the CAM entry tables following the procedure below.
1. Check that the ADSBSY bit in ETNAnTSU_ADSBSY is cleared to 0.
2. Set the upper 32 bits of the MAC addresses by ETNAnTSU_ADRH0 to ETNAnTSU_ADRH31.
3. Set the lower 16 bits of the MAC addresses by ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31.

## 21.3.46 ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31 — CAM Entry Table 0L to 31L Registers

ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31 are entry tables referred to by the CAM in reception. Each of these registers sets the lower 16 bits of the 48-bit MAC address. Maximum 32 entries of MAC addresses can be registered. The CAM entry table setting must not be changed while the transmitting and receiving functions are enabled.

**Access:** This register can be read/written in 32-bit units.

**Address:**

| | |
|---|---|
| ETNAnTSU_ADRL0: <ETNAn_base> + $0904_H$ | ETNAnTSU_ADRL1: <ETNAn_base> + $090C_H$ |
| ETNAnTSU_ADRL2: <ETNAn_base> + $0914_H$ | ETNAnTSU_ADRL3: <ETNAn_base> + $091C_H$ |
| ETNAnTSU_ADRL4: <ETNAn_base> + $0924_H$ | ETNAnTSU_ADRL5: <ETNAn_base> + $092C_H$ |
| ETNAnTSU_ADRL6: <ETNAn_base> + $0934_H$ | ETNAnTSU_ADRL7: <ETNAn_base> + $093C_H$ |
| ETNAnTSU_ADRL8: <ETNAn_base> + $0944_H$ | ETNAnTSU_ADRL9: <ETNAn_base> + $094C_H$ |
| ETNAnTSU_ADRL10: <ETNAn_base> + $0954_H$ | ETNAnTSU_ADRL11: <ETNAn_base> + $095C_H$ |
| ETNAnTSU_ADRL12: <ETNAn_base> + $0964_H$ | ETNAnTSU_ADRL13: <ETNAn_base> + $096C_H$ |
| ETNAnTSU_ADRL14: <ETNAn_base> + $0974_H$ | ETNAnTSU_ADRL15: <ETNAn_base> + $097C_H$ |
| ETNAnTSU_ADRL16: <ETNAn_base> + $0984_H$ | ETNAnTSU_ADRL17: <ETNAn_base> + $098C_H$ |
| ETNAnTSU_ADRL18: <ETNAn_base> + $0994_H$ | ETNAnTSU_ADRL19: <ETNAn_base> + $099C_H$ |
| ETNAnTSU_ADRL20: <ETNAn_base> + $09A4_H$ | ETNAnTSU_ADRL21: <ETNAn_base> + $09AC_H$ |
| ETNAnTSU_ADRL22: <ETNAn_base> + $09B4_H$ | ETNAnTSU_ADRL23: <ETNAn_base> + $09BC_H$ |
| ETNAnTSU_ADRL24: <ETNAn_base> + $09C4_H$ | ETNAnTSU_ADRL25: <ETNAn_base> + $09CC_H$ |
| ETNAnTSU_ADRL26: <ETNAn_base> + $09D4_H$ | ETNAnTSU_ADRL27: <ETNAn_base> + $09DC_H$ |
| ETNAnTSU_ADRL28: <ETNAn_base> + $09E4_H$ | ETNAnTSU_ADRL29: <ETNAn_base> + $09EC_H$ |
| ETNAnTSU_ADRL30: <ETNAn_base> + $09F4_H$ | ETNAnTSU_ADRL31: <ETNAn_base> + $09FC_H$ |

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADRLi[15:0] (i = 0 to 31) | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.54 ETNAnTSU_ADRLn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | ADRL i[15:0]<br>(i = 0 to 31) | MAC Address Bits<br>These bits set the lower 16 bits of the MAC address.<br>When the MAC address is 01-23-45-67-89-AB (displayed in hexadecimal), set $0000\ 89AB_H$ in this register. |

**Note:** Set the CAM entry tables following the procedure below.
1. Check that the ADSBSY bit in ETNAnTSU_ADSBSY is cleared to 0.
2. Set the upper 32 bits of the MAC addresses by ETNAnTSU_ADRH0 to ETNAnTSU_ADRH31.
3. Set the lower 16 bits of the MAC addresses by ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31.

### 21.3.47 ETNAnTXNLCRm — Transmit Frame Counter Register (Port m) (Normal Transmission Only)

ETNAnTXNLCRm is a 32-bit counter indicating the number of frames successfully transmitted in the MACm. When the value in this register reaches FFFF $FFFF_H$, count-up is halted. The counter is cleared to 0 by reading from this register. This register cannot be written to.

**Access:** This register can be read in 32-bit units.

**Address:** ETNAnTXNLCR0: <ETNAn_base> + $0880_H$
ETNAnTXNLCR1: <ETNAn_base> + $08A0_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NTCm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NTCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.55 ETNAnTXNLCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | NTCm[31:0] | Port m Transmit Frame Counter Bits<br>These bits indicate the number of frames successfully transmitted. |

### 21.3.48 ETNAnTXALCRm — Transmit Frame Counter Register (Port m) (Normal and Erroneous Transmission)

ETNAnTXALCRm is a 32-bit counter indicating the number of frames transmitted in the MACm, including the number of frames erroneously transmitted. When the value in this register reaches FFFF FFFF$_H$, count-up is halted. The counter is cleared to 0 by reading from this register. This register cannot be written to.

**Access:** This register can be read in 32-bit units.

**Address:** ETNAnTXALCR0: <ETNAn_base> + 0884$_H$
ETNAnTXALCR1: <ETNAn_base> + 08A4$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.56 ETNAnTXALCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TCm[31:0] | Port m Transmit Frame Counter Bits<br>These bits indicate the number of frames successfully transmitted and erroneously transmitted. |

### 21.3.49 ETNAnRXNLCRm — Receive Frame Counter Register (Port m) (Normal Reception Only)

ETNAnRXNLCRm is a 32-bit counter indicating the number of frames successfully received in the MACm. When the value in this register reaches FFFF $FFFF_H$, count-up is halted. The counter is cleared to 0 by reading from this register. This register cannot be written to. If a frame is discarded in the TSU block by the comparison with the CAM entry, it is not counted.

**Access:** This register can be read in 32-bit units.

**Address:** ETNAnRXNLCR0: <ETNAn_base> + $0888_H$
ETNAnRXNLCR1: <ETNAn_base> + $08A8_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NRCm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NRCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.57 ETNAnRXNLCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | NRCm[31:0] | Port m Receive Frame Counter Bits<br>These bits indicate the number of frames successfully received. |

### 21.3.50 ETNAnRXALCRm — Receive Frame Counter Register (Port m) (Normal and Erroneous Reception)

ETNAnRXALCRm is a 32-bit counter indicating the number of frames received in the MACm, including the number of frames erroneously received. When the value in this register reaches FFFF $FFFF_H$, count-up is halted. The counter is cleared to 0 by reading from this register. This register cannot be written to. If a frame is discarded in the TSU block by the comparison with the CAM entry, it is not counted.

**Access:** This register can be read in 32-bit units.

**Address:** ETNAnRXALCR0: <ETNAn_base> + $088C_H$
ETNAnRXALCR1: <ETNAn_base> + $08AC_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RCm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.58 ETNAnRXALCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RCm[31:0] | Port m Receive Frame Counter Bits<br>These bits indicate the number of frames successfully received and erroneously received. |

## 21.3.51 ETNAnEDSRm — E-DMAC Start Register m

ETNAnEDSRm specifies activation of the transmitting unit and receiving unit of the E-DMACm. ENTm and ENRm can only be written to 1. The method to clear ENTm and ENRm is reset only.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnEDSR0: <ETNAn_base> + $0000_H$
ETNAnEDSR1: <ETNAn_base> + $1000_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | GPOm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ENTm | ENRm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 21.59 ETNAnEDSRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 16 | GPOm | Select the transfer mode<br>0: GPOm = 0 (MII)<br>1: GPOm = 1 (RMII) |
| 15 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | ENTm | E-DMAC Transmitting Unit Start<br>0: Stops the E-DMACm transmitting unit<br>1: Starts the E-DMACm transmitting unit |
| 0 | ENRm | E-DMAC Receiving Unit Start<br>0: Stops the E-DMACm receiving unit<br>1: Starts the E-DMACm receiving unit |

### 21.3.52 ETNAnEDMRm — E-DMAC Mode Register m

ETNAnEDMRm is a 32-bit readable/writable register that specifies E-DMACm resetting and the transmit/receive descriptor length. This register is to be set before the transmitting or receiving function is enabled (before the TRm[1:0] bit in ETNAnEDTRRm or the RRm bit in ETNAnEDRRRm is set to 1). However, the SWRRm and SWRTm bits can be written to even after the transmitting or receiving function is enabled. If a software reset is executed with this register during data transmission, abnormal data may be transmitted on the line. Execute a software reset with this register before specifying the transmit/receive descriptor length or modifying the settings of ETNAnTDLARm, ETNAnRDLARm, and so forth, the setting of ETNAnECMRm (MACm mode register), and the settings of registers related to the E-DMACm and MACm operation.

To execute a software reset with this register, 1 must be written to both the SWRTm and SWRRm bits simultaneously. Writing 1 to the SWRTm and SWRRm bits initializes the MACm registers and E-DMACm registers, except for ETNAnTDLARm, ETNAnRDLARm, and ETNAnRMFCRm of the E-DMACm. The TSU registers (registers whose names are prefixed with ETNAnTSU_) are not initialized. Writing 1 to the SWRTm and SWRRm bits in ETNAnEDMRm initializes the registers related to the E-DMACm and MACm, whereas, writing 1 to the SWRTm and SWRRm bits in ETNAnEDMRm initializes the registers related to the E-DMACm and MACm. Note that during the period a software reset is issued (for 1200 cycles of the HCLK), accesses to all Ethernet-related registers are prohibited.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnEDMR0: <ETNAn_base> + $0400_H$
ETNAnEDMR1: <ETNAn_base> + $1400_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | DEm | DLm[1:0] | | — | — | SWRTm | SWRRm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R | R/W | R/W |

**Table 21.60 ETNAnEDMRm register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 | DEm | Transmit/Receive Frame Endian<br>Sets the endian mode for DMA transfer of frame data between the transmit/receive FIFO and transmit/receive buffer.<br>0: Big endian (longword access)<br>1: Little endian (longword access) |
| 5, 4 | DLm[1:0] | Transmit/Receive Descriptor Length<br>These bits specify the descriptor length. (See Descriptors and Descriptor List.)<br>00: 16 bytes<br>01: 32 bytes<br>10: 64 bytes<br>11: Setting prohibited |

**Table 21.60 ETNAnEDMRm register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3, 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SWRTm | Software Reset of Transmit FIFO Controller<br>[Writing]<br>0: Disabled<br>1: Software reset started<br>[Reading]<br>0: Software reset not executed (or completed)<br>1: Software reset being executed<br>Software reset takes the time about 512 to 1200 HCLK cycles. When Software reset was finished, this bit is cleared automatically.<br>Note: When ENTm of the ETNAnEDSRm register is 0, Software Reset does not work however this bit was set.<br>“Read and Write operation of registers mapped among 0000 $0400_H$ to 0000 $07FF_H$ are canceled during this bit is set. |
| 0 | SWRRm | Software Reset of Receive FIFO Controller<br>[Writing]<br>0: Disabled<br>1: Software reset started<br>[Reading]<br>0: Software reset not executed (or completed)<br>1: Software reset being executed<br>Software reset takes the time about 512 to 1200 HCLK cycles. When Software reset was finished, this bit is cleared automatically.<br>Note: When ENRm of the ETNAnEDSRm register is 0, Software Reset does not work however this bit was set.<br>“Read and Write operation of registers mapped among 0000 $0400_H$ to 0000 $07FF_H$ are canceled during this bit is set. |

## 21.3.53 ETNAnEDTRRm — E-DMACm Transmit Request Register m

ETNAnEDTRRm is a 32-bit readable/writable register that issues transmit directives to the E-DMACm. After writing 11 to bits TRm[1:0] in this register, the E-DMACm reads the transmit descriptor at the address specified by ETNAnTDLARm. If the TACT bit of this transmit descriptor is set to 1 (valid), transmit DMA transfer by the E-DMACm starts. When DMA transfer based on the first transmit descriptor is completed, the E-DMACm reads the next transmit descriptor. If the TACT bit of that transmit descriptor is set to 1 (valid), the E-DMACm continues transmit DMA operation. If the TACT bit of a transmit descriptor is cleared to 0 (invalid), the E-DMACm clears bits TRm[1:0] and stops transmit DMA operation.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnEDTRR0: <ETNAn_base> + $0408_H$
ETNAnEDTRR1: <ETNAn_base> + $1408_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TRm[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 21.61 ETNAnEDTRRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | TRm[1:0] | Transmit Request<br>00, 01, 10: Transmission-halted state<br>If 00, 01, or 10 is written to these bits, the E-DMACm stops DMA transfer of the currently processed transmit descriptor, reads the next transmit descriptor, and then clears these bits. (Write-back is completed for the valid transmit descriptors that have been detected up till then.)<br>The E-DMACm clears these bits when transmit descriptor empty occurs, or transmission of a transmit descriptor has completed. (Write-back is completed for the valid transmit descriptors that have been detected up till then.)<br>11: Transmit DMA operation by E-DMACm<br>After writing 11 to these bits, the E-DMACm starts reading a transmit descriptor. |

## 21.3.54 ETNAnEDRRRm — E-DMACm Receive Request Register m

ETNAnEDRRRm is a 32-bit readable/writable register that issues receive directives to the E-DMACm. After writing 1 to the RRm bit in this register, the E-DMACm reads the receive descriptor at the address specified by ETNAnRDLARm. If the RACT bit of this receive descriptor is set to 1 (valid), and the receive FIFO holds a receive frame, the E-DMACm starts receive DMA transfer. When DMA transfer based on the first receive descriptor is completed, the E-DMACm reads the next receive descriptor. If the RACT bit of that receive descriptor is set to 1 (valid), the E-DMACm continues receive DMA operation. However, if the receive FIFO holds no receive data, the E-DMACm places receive DMA operation in the standby state. If the RACT bit of the receive descriptor is cleared to 0 (invalid), the E-DMACm clears the RRm bit and stops receive DMAC operation.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnEDRRR0: <ETNAn_base> + $0410_H$
ETNAnEDRRR1: <ETNAn_base> + $1410_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RRm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 21.62 ETNAnEDTRRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | RRm | Receive Request<br>0: Receiving function is disabled[*1]<br>If 0 is written to this bit, the E-DMACm stops receive operation after DMA transfer of one frame has completed and then clears this bit.<br>The E-DMACm clears this bit when receive descriptor empty occurs.<br>1: Receive descriptor is read, and the E-DMACm is ready to receive |

Note 1. If the receiving function is disabled during frame reception, write-back is not performed successfully to the receive descriptor. Following pointers to read a receive descriptor become abnormal and the E-DMACm cannot operate successfully. In this case, to make E-DMACm reception enabled again, execute a software reset by the SWRTm and SWRRm bits in ETNAnEDMRm. To disable the E-DMACm receiving function without executing a software reset, specify the REm bit in ETNAnECMRm. Next, after the E-DMACm has completed the reception and write-back to the receive descriptor has been confirmed, disable the receiving function using this register.

## 21.3.55 ETNAnTDLARm — Transmit Descriptor List Start Address Register m

ETNAnTDLARm is a 32-bit readable/writable register that specifies the start address of the transmit descriptor list. Descriptors have a boundary configuration in accordance with the descriptor length indicated by the DLm[1:0] bits in ETNAnEDMRm. This register must not be modified during transmission. Modifications to this register should only be made in the transmission-halted state specified by bits TRm[1:0] (= 00) in the E-DMACm transmit request register (ETNAnEDTRRm).

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTDLAR0: <ETNAn_base> + $0010_H$
ETNAnTDLAR1: <ETNAn_base> + $1010_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TDLAm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TDLAm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.63 ETNAnTDLARm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TDLAm[31:0] | Transmit Descriptor Start Address<br>The lower bits are set according to the specified descriptor length.<br>16-byte boundary: TDLA[3:0] = 0000<br>32-byte boundary: TDLA[4:0] = 00000<br>64-byte boundary: TDLA[5:0] = 000000 |

### 21.3.56 ETNAnRDLARm — Receive Descriptor List Start Address Register m

ETNAnRDLARm is a 32-bit readable/writable register that specifies the start address of the receive descriptor list. Descriptors have a boundary configuration in accordance with the descriptor length indicated by the DLm[1:0] bits in ETNAnEDMRm. This register must not be modified during reception. Modifications to this register should only be made while reception is disabled by the RRm bit (= 0) in the E-DMACm receive request register (ETNAnEDRRRm).

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnRDLAR0: <ETNAn_base> + $0030_H$
ETNAnRDLAR1: <ETNAn_base> + $1030_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDLAm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDLAm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.64 ETNAnRDLARm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RDLAm[31:0] | Receive Descriptor Start Address<br>The lower bits are set according to the specified descriptor length.<br>16-byte boundary: RDLA[3:0] = 0000<br>32-byte boundary: RDLA[4:0] = 00000<br>64-byte boundary: RDLA[5:0] = 000000 |

## 21.3.57 ETNAnEESRm — E-MAC/E-DMAC Status Register m

ETNAnEESRm is a 32-bit readable/writable register that shows communications status information on the E-DMACm in combination with the MACm. The information in this register is reported in the form of interrupt sources. Individual bits are cleared by writing 1 (however, bit 22 (ECIm) is a read-only bit that is not cleared by writing 1) and are not affected by writing 0. Each interrupt source can also be masked by means of the corresponding bit in the MACm/E-DMAC status interrupt permission register (ETNAnEESIPRm).

The interrupts generated by this status register are HINT_0 for port 0 and HINT_1 for port 1. For interrupt priorities, see **Section 6, Interrupt Controller (INTC).**

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnEESR0: <ETNAn_base> + $0428_H$
ETNAnEESR1: <ETNAn_base> + $1428_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TWBm[1:0] | | TCm[1] | — | ROCm | — | RABTm | RFCOFm | — | ECIm | TCm[0] | TDEm | — | FRm | RDEm | RFOFm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R | R/W | R | R/W | R/W | R | R | R/W | R/W | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | DLCm | CDm | TROm | RMAFm | — | — | RRFm | RTLFm | RTSFm | PREm | CERFm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 21.65 ETNAnEESRm register contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | TWBm[1:0] | Write-Back Complete<br>Indicates that write-back from the E-DMACm to the corresponding descriptor after frame transmission has completed. This operation is enabled only when the TWBI bit in the transmit descriptor that includes the end of the transmit frame is set to 1.<br>00: Write-back has not completed, or no transmission directive<br>11: Write-back has completed<br>Others: Setting disabled |
| 29 | TCm[1] | Frame Transmission Complete<br>Indicates, in combination with the TCm[0] bit, that all the data specified by the transmit descriptor has been transmitted from the MACm. This bit is set to 1 on assuming the completion of transmission. This is when transmission of one frame is completed and the transmit descriptor valid bit (TACT) of the next transmit descriptor not being set in single-frame/single-descriptor operation or when the last data of a frame has been transmitted and the transmit descriptor valid bit (TACT) of the next descriptor not being set in multi-buffer frame processing based on single-frame/multi-descriptor operation. After frame transmission has completed, the E-DMACm writes the transmission status back to the relevant descriptor.<br>TCm[1:0]<br>00: Transmission has not completed, or no transmission directive<br>11: Transmission has completed<br>Others: Setting disabled |
| 28 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 27 | ROCm | Receive Overflow Frame Write-Back Complete<br>0: Write-back has not completed for the frame causing receive overflow<br>1: Write-back has completed for the frame causing receive overflow |

Table 21.65 ETNAnEESRm register contents (2/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 26 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 25 | RABTm | Receive Abort Detect<br>Indicates that the MACm aborts receiving a frame because of failures during frame reception.<br>0: Frame reception has not been aborted or no reception directive<br>1: Frame reception has been aborted |
| 24 | RFCOFm | Receive Frame Counter Overflow<br>Indicates that the frame counter in the receive FIFO has overflowed.<br>0: Receive frame counter has not overflowed<br>1: Receive frame counter has overflowed |
| 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22 | ECIm | MACm Status Register Source<br>This bit is a read-only bit. When the source of an ETNAnECSRm interrupt is cleared, this bit is also cleared.<br>0: MACm status interrupt source has not been detected<br>1: MACm status interrupt source has been detected |
| 21 | TCm[0] | Frame Transmission Complete<br>Indicates, in combination with the TCm[1] bit, that all the data specified by the transmit descriptor has been transmitted from the MACm. For details, see the description of the TCm[1] bit. |
| 20 | TDEm | Transmit Descriptor Empty<br>Indicates that the transmit descriptor valid bit (TACT) of a transmit descriptor read by the E-DMACm is not set if the previous descriptor does not represent the end of a frame in multi-buffer frame processing based on single-frame/multi-descriptor operation. As a result, an incomplete frame may be sent.<br>0: Transmit descriptor active bit TACT = 1 detected<br>1: Transmit descriptor active bit TACT = 0 detected<br>When transmit descriptor empty (TDE = 1) occurs, execute a software reset and initiate transmission. In this case, transmission starts from the address that is stored in the transmit descriptor list start address register (ETNAnTDLARm). |
| 19 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 18 | FRm | Frame Reception<br>Indicates that a frame has been received and the receive descriptor has been updated. This bit is set to 1 each time a frame is received.<br>0: Frame has not been received<br>1: Frame has been received |
| 17 | RDEm | Receive Descriptor Empty<br>Indicates that the RACT bit of a receive descriptor read by the E-DMACm for receive DMA operation is cleared to 0 (invalid).<br>When receive descriptor empty (RDE = 1) occurs, reception can be resumed by setting the RACT bit (cleared to 0) of the receive descriptor to 1 and then writing 1 to the RRm bit in ETNAnEDRRRm.<br>0: Receive descriptor active bit RACT = 1 detected<br>1: Receive descriptor active bit RACT = 0 detected |
| 16 | RFOFm | Receive FIFO Overflow<br>Indicates that the receive FIFO has overflowed during frame reception.<br>0: Overflow has not occurred<br>1: Overflow has occurred |
| 15 to 11 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 10 | DLCm | Detect Loss of Carrier<br>Indicates that loss of the carrier has been detected during frame transmission.<br>0: Loss of carrier has not been detected<br>1: Loss of carrier has been detected |

Table 21.65 ETNAnEESRm register contents (3/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 9 | CDm | Delayed Collision Detect<br>Indicates that a delayed collision has been detected during frame transmission.<br>0: Delayed collision has not been detected<br>1: Delayed collision has been detected |
| 8 | TROm | Transmit Retry Over<br>Indicates that a retry-over condition has occurred during frame transmission. Total 16 transmission retries including 15 retries based on the back-off algorithm have failed after the MACm transmission starts.<br>0: Transmit retry-over condition not detected<br>1: Transmit retry-over condition detected |
| 7 | RMAFm | Receive Multicast Address Frame<br>0: Multicast address frame has not been received<br>1: Multicast address frame has been received |
| 6, 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | RRFm | Receive Residual-Bit Frame<br>0: Residual-bit frame has not been received<br>1: Residual-bit frame has been received |
| 3 | RTLFm | Receive Too-Long Frame<br>Indicates that a frame whose byte size exceeds the upper limit for the receive frame length set by ETNAnRFLRm has been received.<br>0: Too-long frame has not been received<br>1: Too-long frame has been received |
| 2 | RTSFm | Receive Too-Short Frame<br>Indicates that a frame of fewer than 64 bytes has been received.<br>0: Too-short frame has not been received<br>1: Too-short frame has been received |
| 1 | PREm | PHY-LSI Receive Error(ETNAmRXER)<br>0: PHY-LSI receive error or carrier extension error has not been detected<br>1: PHY-LSI receive error or carrier extension error has been detected |
| 0 | CERFm | CRC Error on Received Frame<br>0: CRC error has not been detected<br>1: CRC error has been detected |

## 21.3.58 ETNAnEESIPRm — E-MAC/E-DMAC Status Interrupt Permission Register m

ETNAnEESIPRm is a 32-bit readable/writable register that enables interrupts corresponding to individual bits in the MACm/E-DMACm status register (ETNAnEESRm) in combination with HDIEm bit in ETNAnHDMIERm. An interrupt is enabled by writing 1 to the corresponding bit.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnEESIPR0: <ETNAn_base> + $0430_H$
ETNAnEESIPR1: <ETNAn_base> + $1430_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TWB1nIP | TWB0mIP | TC1nIP | — | ROCmIP | — | RABTmIP | RFCOFmIP | — | ECImIP | TC0mIP | TDEmIP | — | FRmIP | RDEmIP | RFOFmIP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R | R/W | R | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | DLCmIP | CDmIP | TROmIP | RMAFmIP | — | — | RRFmIP | RTLFmIP | RTSFmIP | PREmIP | CERFmIP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 21.66 ETNAnEESIPRm register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | TWB1nIP | Write-Back Complete Interrupt Enable<br>0: Write-back complete interrupt is disabled<br>1: Write-back complete interrupt is enabled |
| 30 | TWB0mIP | Write-Back Complete Interrupt Enable<br>0: Write-back complete interrupt is disabled<br>1: Write-back complete interrupt is enabled |
| 29 | TC1nIP | Frame Transmission Complete Interrupt Enable<br>0: Frame transmission complete interrupt is disabled<br>1: Frame transmission complete interrupt is enabled |
| 28 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 27 | ROCmIP | Receive Overflow Frame Write-Back Complete Interrupt Enable<br>0: Receive overflow frame write-back complete interrupt is disabled<br>1: Receive overflow frame write-back complete interrupt is enabled |
| 26 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 25 | RABTmIP | Receive Abort Detect Interrupt Enable<br>0: Receive abort detect interrupt is disabled<br>1: Receive abort detect interrupt is enabled |
| 24 | RFCOFmIP | Receive Frame Counter Overflow Interrupt Enable<br>0: Receive frame counter overflow interrupt is disabled<br>1: Receive frame counter overflow interrupt is enabled |
| 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22 | ECImIP | MACm Status Register Source Interrupt Enable<br>0: MACm status interrupt is disabled<br>1: MACm status interrupt is enabled |
| 21 | TC0mIP | Frame Transmission Complete Interrupt Enable<br>0: Frame transmission complete interrupt is disabled<br>1: Frame transmission complete interrupt is enabled |

**Table 21.66 ETNAnEESIPRm register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 20 | TDEmIP | Transmit Descriptor Empty Interrupt Enable<br>0: Transmit descriptor empty interrupt is disabled<br>1: Transmit descriptor empty interrupt is enabled |
| 19 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 18 | FRmIP | Frame Reception Interrupt Enable<br>0: Frame reception interrupt is disabled<br>1: Frame reception interrupt is enabled |
| 17 | RDEmIP | Receive Descriptor Empty Interrupt Enable<br>0: Receive descriptor empty interrupt is disabled<br>1: Receive descriptor empty interrupt is enabled |
| 16 | RFOFmIP | Receive FIFO Overflow Interrupt Enable<br>0: Overflow interrupt is disabled<br>1: Overflow interrupt is enabled |
| 15 to 11 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 10 | DLCmIP | Detect Loss of Carrier Interrupt Enable<br>0: Detect loss of carrier interrupt is disabled<br>1: Detect loss of carrier interrupt is enabled |
| 9 | CDmIP | Delayed Collision Detect Interrupt Enable<br>0: Delayed collision detect interrupt is disabled<br>1: Delayed collision detect interrupt is enabled |
| 8 | TROmIP | Transmit Retry Over Interrupt Enable<br>0: Transmit retry over interrupt is disabled<br>1: Transmit retry over interrupt is enabled |
| 7 | RMAFmIP | Receive Multicast Address Frame Interrupt Enable<br>0: Receive multicast address frame interrupt is disabled<br>1: Receive multicast address frame interrupt is enabled |
| 6, 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | RRFmIP | Receive Residual-Bit Frame Interrupt Enable<br>0: Receive residual-bit frame interrupt is disabled<br>1: Receive residual-bit frame interrupt is enabled |
| 3 | RTLFmIP | Receive Too-Long Frame Interrupt Enable<br>0: Receive too-long frame interrupt is disabled<br>1: Receive too-long frame interrupt is enabled |
| 2 | RTSFmIP | Receive Too-Short Frame Interrupt Enable<br>0: Receive too-short frame interrupt is disabled<br>1: Receive too-short frame interrupt is enabled |
| 1 | PREmIP | PHY-LSI Receive Error Interrupt Enable<br>0: PHY-LSI receive error interrupt is disabled<br>1: PHY-LSI receive error interrupt is enabled |
| 0 | CERFmIP | CRC Error on Received Frame Interrupt Enable<br>0: CRC error interrupt is disabled<br>1: CRC error interrupt is enabled |

## 21.3.59 ETNAnTRSCERm — Transmit/Receive Status Copy Enable Register m

ETNAnTRSCERm specifies whether the information for the transmit and receive state reported by bits 25, and 7 and 4 to 0 in the MACm/E-DMACm status register (ETNAnEESRm) is to be reflected in the RFE bit of the corresponding descriptor. The bits in this register correspond to bits 25, 7 and bit 4 to 0 in ETNAnEESRm. When a bit is cleared to 0, and the receive status (bits 25, 7 and 5 to 0 in ETNAnEESRm) is reflected in the RFE bit of the receive descriptor. In this case, the state of a status bit set to 1 is reflected as the RFE bit set to 1. When a bit is set to 1, the occurrence of the corresponding source is not reflected in the descriptor. After this Ethernet module is reset, all bits are cleared to 0.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTRSCER0: <ETNAn_base> + $0438_H$
ETNAnTRSCER1: <ETNAn_base> + $1438_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RABTmCE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RMAFmCE | — | — | RRFmCE | RTLFmCE | RTSFmCE | PREmCE | CERFmCE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 21.67 ETNAnTRSCERm register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 16 | RABTmCE | RABTm Bit Copy Directive<br>0: Reflects the RABTm bit status in the RFE bit of the receive descriptor<br>1: Occurrence of the corresponding source is not reflected in the RFE bit of the receive descriptor |
| 15 to 8 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 7 | RMAFmCE | RMAFm Bit Copy Directive<br>0: Reflects the RMAFm bit status in the RFE bit of the receive descriptor<br>1: Occurrence of the corresponding source is not reflected in the RFE bit of the receive descriptor |
| 6, 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | RRFmCE | RRFm Bit Copy Directive<br>0: Reflects the RRFm bit status in the RFE bit of the receive descriptor<br>1: Occurrence of the corresponding source is not reflected in the RFE bit of the receive descriptor |
| 3 | RTLFmCE | RTLFm Bit Copy Directive<br>0: Reflects the RTLFm bit status in the RFE bit of the receive descriptor<br>1: Occurrence of the corresponding source is not reflected in the RFE bit of the receive descriptor |
| 2 | RTSFmCE | RTSFm Bit Copy Directive<br>0: Reflects the RTSFm bit status in the RFE bit of the receive descriptor<br>1: Occurrence of the corresponding source is not reflected in the RFE bit of the receive descriptor |

**Table 21.67 ETNAnTRSCERm register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | PREmCE | PREm Bit Copy Directive<br>0: Reflects the PREm bit status in the RFE bit of the receive descriptor<br>1: Occurrence of the corresponding source is not reflected in the RFE bit of the receive descriptor |
| 0 | CERFmCE | CERFm Bit Copy Directive<br>0: Reflects the CERFm bit status in the RFE bit of the receive descriptor<br>1: Occurrence of the corresponding source is not reflected in the RFE bit of the receive descriptor |

## 21.3.60 ETNAnRMFCRm — Receive Missed-Frame Counter Register m

ETNAnRMFCRm is a 16-bit counter that indicates the number of frames that could not be saved in the receive buffer and so were discarded during reception. When the receive FIFO overflows, the receive frames in the FIFO are discarded. The number of frames discarded at this time is counted. When the value in this register reaches $FFFF_H$, count-up is halted. Clear the counter by writing $0000_H$ in this register.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnRMFCR0: <ETNAn_base> + $0440_H$
ETNAnRMFCR1: <ETNAn_base> + $1440_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MFCm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.68 ETNAnRMFCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 to 0 | MFCm[15:0] | Missed-Frame Counter<br>These bits indicate the number of frames that are discarded and not transferred to the receive buffer during reception. |

## 21.3.61 ETNAnFDRm — FIFO Depth Register m

ETNAnFDRm is a 32-bit readable/writable register that specifies the sizes of the transmit and receive FIFOs.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnFDR0: <ETNAn_base> + $0450_H$
ETNAnFDR1: <ETNAn_base> + $1450_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | TFDm[4:0] | | | | | RFDm[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.69 ETNAnFDRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 12 to 8 | TFDm[4:0] | Transmit FIFO Size<br>Specifies 256 bytes to 2 Kbytes in 256-byte units as the size of the transmit FIFO whose maximum size is 8 Kbytes. The setting must not be changed after transmission/reception has started.<br>$00_H$: 256 bytes<br>$01_H$: 512 bytes<br>: :<br>$07_H$: 2048 bytes<br>Settings from $08_H$ to $FF_H$ are prohibited. |
| 7 to 0 | RFDm[7:0] | Receive FIFO Size<br>Specifies 256 bytes to 12 Kbytes in 256-byte units as the size of the receive FIFO whose maximum size is 64 Kbytes. The setting must not be changed after transmission/reception has started.<br>$00_H$: 256 bytes<br>$01_H$: 512 bytes<br>$02_H$: 768 bytes<br>$03_H$:1024 bytes<br>$04_H$: 1280 bytes<br>: :<br>$07_H$: 2048 bytes<br>Settings from $08_H$ to $FF_H$ are prohibited. |

**NOTE**

The transmit frame which has the size beyond TFDm configuration can not be transmitted. TFDm must be specified for the size larger than or equal to maximum frame size to transmit.

## 21.3.62 ETNAnRMCRm — Receiving Method Control Register m

ETNAnRMCRm is a 32-bit readable/writable register that specifies the control method for the REm bit in ETNAnECMRm while a frame is received. This register must be set during the receiving-halted state.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnRMCR0: <ETNAn_base> + $0458_H$
ETNAnRMCR1: <ETNAn_base> + $1458_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RNCm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 21.70 ETNAnRMCRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | RNCm | Receive Enable Control<br>Sets whether to continue frame reception.<br>0: Upon completion of reception of one frame, the E-DMACm writes the receive status to the descriptor and clears the RRm bit in ETNAnEDRRRm to 0.<br>1: Upon completion of reception of one frame, the E-DMACm writes (writes back) the receive status to the descriptor. In addition, the E-DMACm reads the next descriptor and prepares for reception of the next frame. |

### 21.3.63 ETNAnRDFARm — Receive Descriptor Fetch Address Register m

ETNAnRDFARm stores the descriptor start address that is required when the E-DMACm fetches descriptor information from the receive descriptor. Which receive descriptor information is used for processing by the E-DMACm can be recognized by monitoring addresses displayed in this register. The address from which the E-DMACm is actually fetching a descriptor may be different from the value read from this register. In the initial setting, set the address of the receive descriptor at which receive processing is to be started.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnRDFAR0: <ETNAn_base> + $0034_H$
ETNAnRDFAR1: <ETNAn_base> + $1034_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDFAm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDFAm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.71 ETNAnRDFARm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RDFAm[31:0] | Receive Descriptor Fetch Address<br>Writing to these bits during the reception is prohibited. |

### 21.3.64 ETNAnRDFXRm — Receive Descriptor Finished Address Register m

ETNAnRDFXRm stores the start address of the receive descriptor for which the E-DMACm has just completed the write-back processing. Up to which receive descriptor has been processed by the E-DMACm can be recognized by monitoring addresses displayed in this register. In the initial setting, set the address of the descriptor immediately before the descriptor that is pointed to by the address in ETNAnRDFARm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnRDFXR0: <ETNAn_base> + $0038_H$
ETNAnRDFXR1: <ETNAn_base> + $1038_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDFXm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDFXm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.72 ETNAnRDFXRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | RDFXm[31:0] | Receive Descriptor Finished Address<br>Writing to these bits during the reception is prohibited. |

## 21.3.65 ETNAnRDFFRm — Receive Descriptor Final Flag Register m

ETNAnRDFFRm indicates whether the receive descriptor for which the E-DMACm has just completed the write-back processing and whose start address is stored in ETNAnRDFXRm is at the end of the receive descriptor queue (descriptor list).

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnRDFFR0: <ETNAn_base> + $003C_H$
ETNAnRDFFR1: <ETNAn_base> + $103C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RDLFm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 21.73 ETNAnRDFFRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | RDLFm | Receive Descriptor Queue Last Flag<br>Indicates whether the receive descriptor for which the E-DMACm has just completed the write-back processing and whose start address is stored in ETNAnRDFXRm is at the end of the receive descriptor queue (descriptor list).<br>0: Not the last descriptor in the receive descriptor queue<br>1: Last descriptor in the receive descriptor queue |

### 21.3.66 ETNAnTDFARm — Transmit Descriptor Fetch Address Register m

ETNAnTDFARm stores the descriptor start address that is required when the E-DMACm fetches descriptor information from the transmit descriptor. Which transmit descriptor information is used for processing by the E-DMACm can be recognized by monitoring addresses displayed in this register. The address from which the E-DMACm is actually fetching a descriptor may be different from the value read from this register. In the initial setting, set the address of the transmit descriptor at which transmit processing is to be started.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTDFAR0: <ETNAn_base> + $0014_H$
ETNAnTDFAR1: <ETNAn_base> + $1014_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TDFAm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TDFAm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.74 ETNAnTDFARm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TDFAm[31:0] | Transmit Descriptor Fetch Address<br>Writing to these bits during transmission is prohibited. |

## 21.3.67 ETNAnTDFXRm — Transmit Descriptor Finished Address Register m

ETNAnTDFXRm stores the start address of the transmit descriptor for which the E-DMACm has just completed the write-back processing. Up to which transmit descriptor has been processed by the E-DMACm can be recognized by monitoring addresses displayed in this register. In the initial setting, set the address of the transmit descriptor immediately before the descriptor that is pointed to by the address in ETNAnTDFARm.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTDFXR0: <ETNAn_base> + $0018_H$
ETNAnTDFXR1: <ETNAn_base> + $1018_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TDFXm[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TDFXm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.75 ETNAnTDFXRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TDFXm[31:0] | Transmit Descriptor Finished Address<br>Writing to these bits during transmission is prohibited. |

## 21.3.68 ETNAnTDFFRm — Transmit Descriptor Final Flag Register m

ETNAnTDFFRm indicates whether the transmit descriptor for which the E-DMACm has just completed the write-back processing and whose start address is stored in ETNAnTDFXRm is at the end of the transmit descriptor queue (descriptor list).

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnTDFFR0: <ETNAn_base> + $001C_H$
ETNAnTDFFR1: <ETNAn_base> + $101C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TDLFm |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 21.76 ETNAnTDFFRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | TDLFm | Transmit Descriptor Queue Last Flag<br>Indicates whether the transmit descriptor for which the E-DMACm has just completed the write-back processing and whose start address is stored in ETNAnTDFXRm is at the end of the transmit descriptor queue (descriptor list).<br>0: Not the last descriptor in the transmit descriptor queue<br>1: Last descriptor in the transmit descriptor queue |

## 21.3.69 ETNAnFCFTRm — Overflow Alert FIFO Threshold Register m

ETNAnFCFTRm is a 32-bit readable/writable register that sets the flow control of the MACm. The threshold can be set by the size of the receive FIFO data (bits RFDm[7:0]) and the number of receive frames (bits RFFm[4:0]).

If the same receive FIFO size as set by the FIFO depth register (ETNAnFDRm) is set when flow control is turned on according to the RFDm setting condition, flow control is turned on with (FIFO data size – 64) bytes. For instance, when the RFDm bits in ETNAnFDRm = 7 and the RFSm bits in this register = 7, flow control is turned on when (2,048 to 64) bytes of data is stored in the receive FIFO. The value set in the RFSm bits in this register should be equal to or less than that set in the RFDm bits in ETNAnFDRm.

Flow control is turned on when either of the setting conditions of bits RFFm[4:0] and bits RFSm[7:0] is satisfied. Flow control is turned off when neither of the conditions is satisfied (release).

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnFCFTR0: <ETNAn_base> + $0468_H$
ETNAnFCFTR1: <ETNAn_base> + $1468_H$

**Value after reset:** 0017 $00FF_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | RFFm[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RFSm[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.77 ETNAnFCFTRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 20 to 16 | RFFm[4:0] | Receive FIFO Overflow Alert Signal Output Threshold<br>$00_H$: When one receive frame has been stored in the receive FIFO<br>$01_H$: When two receive frames have been stored in the receive FIFO<br>: :<br>$16_H$: When 23 receive frames have been stored in the receive FIFO<br>$17_H$: When 24 receive frames have been stored in the receive FIFO<br>$18_H$ to H'1F: Prohibition |
| 15 to 8 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 7 to 0 | RFSm[7:0] | Receive FIFO Overflow Alert Signal Output Threshold<br>$00_H$: When 256 bytes of data is stored in the receive FIFO<br>$01_H$: When 512 bytes of data is stored in the receive FIFO<br>: :<br>$06_H$: When 1,792 bytes of data is stored in the receive FIFO<br>$07_H$: When 2,048 bytes of data is stored in the receive FIFO<br>: :<br>$FF_H$: When 65536 bytes of data is stored in the receive FIFO<br>**NOTE**<br>Don't set this register larger than receive FIFO size(RFDm bits in ETNAnFDRm register) |

## 21.3.70 ETNAnTFSRm — Transmit FIFO Status Register m

ETNAnTFSRm is a 32-bit read only register that shows the status of the transmit FIFO.

**Access:** This register can be read in 32-bit units.

**Address:** ETNAnTFSR0: <ETNAn_base> + $0480_H$
ETNAnTFSR1: <ETNAn_base> + $1480_H$

**Value after reset:** $0200\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TWBRm | — | — | — | — | — | — | — | TFAMm[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TDAMm[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.78 ETNAnTFSRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | When read, the value after reset is read. |
| 26 | TWBRm | Transmit Descriptor write back request |
| 25 to 19 | Reserved | When read, the value after reset is read. |
| 18 to 16 | TFAMm | Transmit FIFO prepared Frame count |
| 15 to 9 | Reserved | When read, the value after reset is read. |
| 8 to 0 | TDAMm | Transmit FIFO used size |

## 21.3.71 ETNAnRFSRm — Receive FIFO Status Register m

ETNAnRFSRm is a 32-bit read only register that shows the status of the receive FIFO.

**Access:** This register can be read in 32-bit units.

**Address:** ETNAnRFSR0: <ETNAn_base> + $0488_H$
ETNAnRFSR1: <ETNAn_base> + $1488_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | RFAMm[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | RDAMm[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 21.79 ETNAnRFSRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | When read, the value after reset is read. |
| 18 to 16 | RFAMm | Receive FIFO prepared Frame count |
| 15 to 9 | Reserved | When read, the value after reset is read. |
| 8 to 0 | RDAMm | Receive FIFO used size |

## 21.3.72 ETNAnRPADIRm — Receive Data Padding Insert Register m

ETNAnRPADIRm is a 32-bit readable/writable register that inserts padding in receive data. When changing the settings of this register, execute a software reset by means of the SWRTm and SWRRm bits in the E-DMACm mode register (ETNAnEDMRm) before making settings again.

**Access:** This register can be read/written in 32-bit units.

**Address:** ETNAnRPADIR0: <ETNAn_base> + $0460_H$
ETNAnRPADIR1: <ETNAn_base> + $1460_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by system reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | PADSm[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PADRm[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 21.80 ETNAnRPADIRm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 20 to 16 | PADSm[4:0] | Padding Size<br>$00_H$: No padding insertion<br>$01_H$: 1-byte insertion<br>: :<br>$1F_H$: 31-byte insertion |
| 15 to 0 | PADRm[15:0] | Padding Slot<br>$0000_H$: Inserts specified size of padding at the first byte<br>$0001_H$: Inserts specified size of padding at the second byte<br>: :<br>$FFFF_H$: Inserts specified size of padding at the 64K byte |

**NOTE**

When PADRm[15:0] is not $0000_H$, the sum of PADRm[1:0] and PADSm[1:0] must be $100_B$ or $000_B$

# 21.4 Operation

## 21.4.1 Limited Reset and Module Stop

As for this module, Limited Reset and Module Stop are possible.
After enforcement of Termination Procedure, please carry out Limited Reset and Module Stop.
There is no enforcement condition of Limited Reset and Module Stop if you carry out Termination Procedure.
See **Section 21.4.38** for Termination Procedure.

## 21.4.2 Basic Frame

The basic frame format defined in IEEE 802.3 consists of the preamble (PA), start frame delimiter (SFD), destination address (DA), source address (SA), type/length field (TYPE/LEN), data field (DATA) and frame check sequence field FCS).

The defined packet size is from 64 bytes (minimum) to 1518 bytes (maximum) excluding the PA and the SFD.

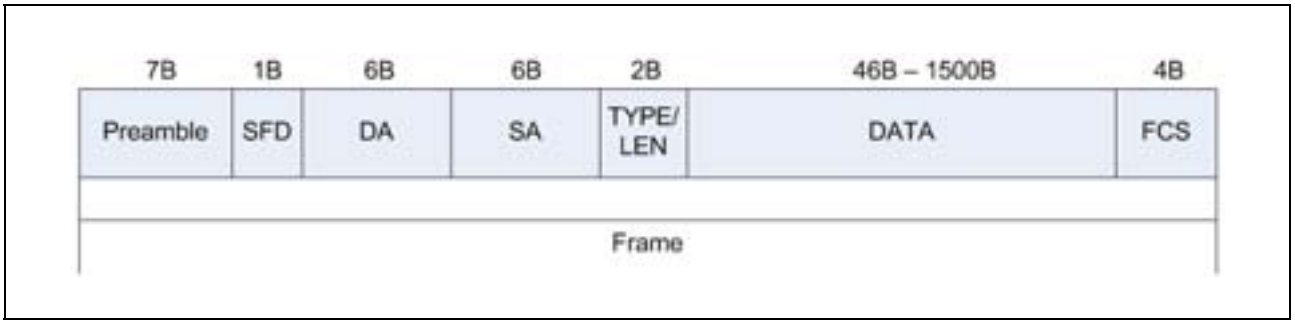



**Figure 21.2 Basic frame structure**

### (1) Preamble and SFD

The preamble and SFD fields consist of a repetition of "$10_B$" for 62 bits followed by "$11_B$" at the end. They represent the start of an Ethernet frame.

### (2) Destination address (DA)

The destination address field indicates the destination MAC address that may be a unicast address, a multicast address or a broadcast address.

### (3) Source address (SA)

The source address field contains the source MAC address.

### (4) Frame type and length (TYPE/LEN)

An Ethernet frame uses this field to specify the protocol type. An IEEE802.3 frame uses this field to indicate the length of the data field.

#### (5) Payload data (DATA)

The payload data field contains the data up to 1500 bytes. The Ethernet controller uses the data in this field for CRC (Cyclic Redundancy Check) calculation for the FCS.

#### (6) Frame check sequence (FCS)

The frame check sequence field is used to write a 32-bit CRC to check the payload data.

### 21.4.3 PAUSE Control Frame

A pause control frame contains 64-byte in a special format . The DA field is fixed to “01-80-C2-00-00-01H”. The Type/Length field contains “$8808_H$” which indicates a control frame. The OP code contains “$0001_H$” which indicates pause control. The parameter field is specified with the time parameter. The unused area following the parameter field is filled with PAD data consisting of zeros.

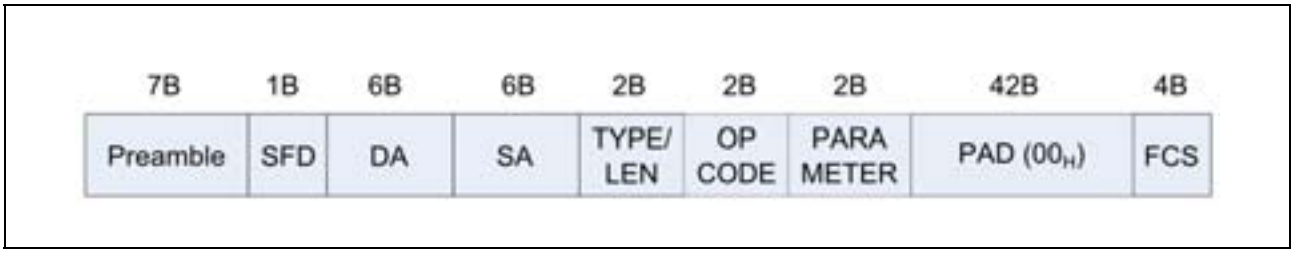



Figure 21.3 Pause control frame structure

### 21.4.4 Magic Packet™

The magic packet™ is a broadcast frame that contains anywhere in the payload 6 times “0xFF” followed by the MAC address of target device repeated 16 times.

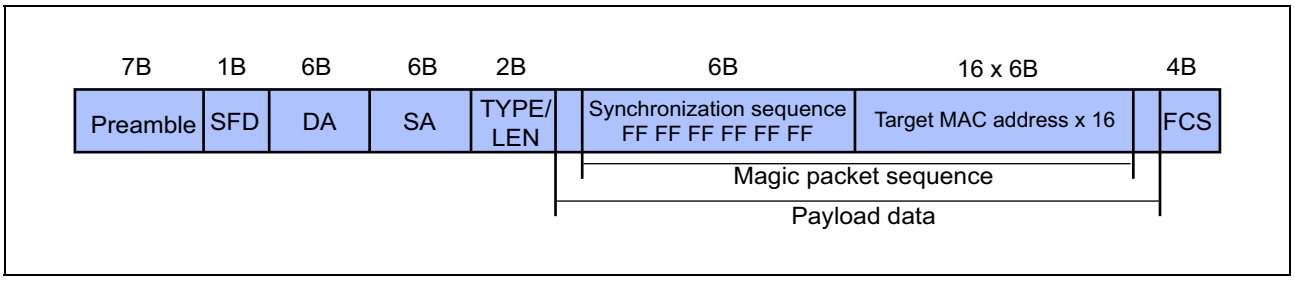



Figure 21.4 Magic packet structure

### 21.4.5 Transmission

The transmission function of the ETNA creates a frame as defined in IEEE802.3 from the data stored in the transmission FIFO through EDMAC transfer and outputs the frame to a PHY device. Upon collision detection, the frame is retransmitted using the back-off algorithm.

### 21.4.6 Transmission Clock

The transmission of ETNA is synchronized with the external transmission clock (ETHmTXCLK).

### 21.4.7 Collision Detection and Retransmission

In half-duplex mode, if a collision within the collision window (512 bits data) is detected before the maximum allowable number of collisions (default = 15) is not exceeded, the ETNA gets the transmission to wait using the random back-off algorithm and retransmits the data.

### 21.4.8 CRC Function

The CRC function generates a CRC code and appends it to the FCS field when ETNA is sending a frame.

### 21.4.9 Transmission Frame Data Padding Function

The specified minimal frame length is 64 bytes. If the data field in a frame is shorter than 46 bytes, padding data (0x00) is added to the data to meet the minimal frame size.

### 21.4.10 Full-/half-duplex Communication

The ETNA supports both full- and half-duplex communication modes. In full-duplex mode, the carrier (CRS) and collisions (COL) are not detected.

### 21.4.11 Inter-packet Gap (IPG)

The inter-packet gap (prescribed in IEEE802.3 standard) between transmit frames can be changed. When changing the IPG setting, adequately check that the respective devices can operate smoothly.

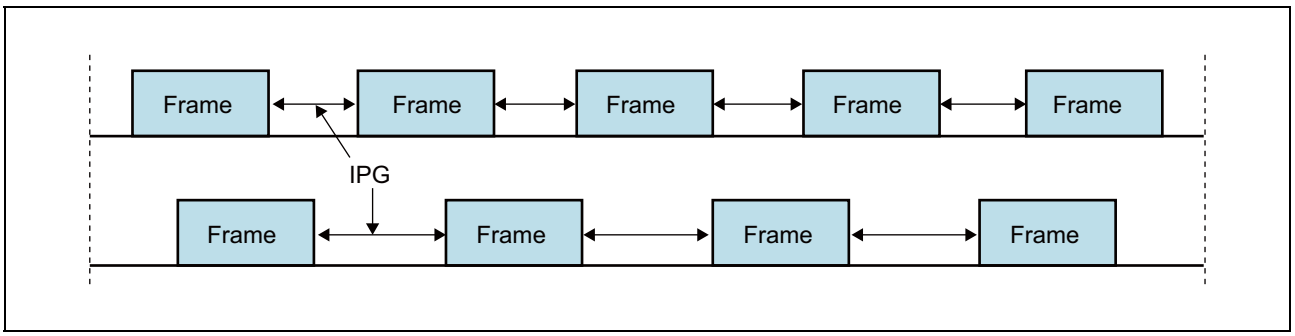



Figure 21.5 Changing IPG and transmission efficiency

### 21.4.12 Reception

The ETNA creates a reception frame by performing SFD detection, destination address check, frame length check, CRC check, etc. on the received data via MII.

### 21.4.13 Reception Clock

The ETNA receives data in synchronization with the reception clock (ETHmRXCLK).

### 21.4.14 Detection of Preamble and SFD

After detection of a preamble and SFD the ETNA starts with the frame reception.

### 21.4.15 CRC Function

The ETNA calculates a 4-byte CRC and compares it with the FCS field of the received frame. The check result is indicated as a status flag in the descriptor after the frame data has been written to the memory. The reception controller reports an error status if a CRC error occurs.

### 21.4.16 Promiscuous Mode

The promiscuous mode can be en/disabled in the ETNA. In promiscuous mode all incoming frames from MII are received. In non-promiscuous mode only the frames that address to the device MAC address, multicast and broadcast frames can be received.

### 21.4.17 Magic Packet Detection

The Magic Packet detection function provides a Wake-On-LAN feature that activates peripheral device connected to a LAN from a host device or other sources. With a Magic Packet reception is performed regardless of the destination address in the frame header. Only in case the 16 times duplicated destination address contained in the Magic Packet Sequence matches the MAC address specified in the device, the CPU will be informed by the Magic Packet detection interrupt and the Wake-On-LAN (ETHmWOL) signal notifies that a Magic Packet has been detected.

### 21.4.18 Flow Control

The flow control function in the ETNA is in compliance with IEEE802.3. The flow control can be applied to both receive and transmit operation.

### 21.4.19 PAUSE Frame Transmission

In full-duplex mode two types of PAUSE frame transmission are available for the flow control:

1. Automatic PAUSE frame transmission
   For receiving frames, PAUSE frames are automatically transmitted when the volume of data written to the reception FIFO reaches the threshold value. The TIME parameter included in the PAUSE frame is set by APR register. The automatic PAUSE frame transmission is repeated until the number of data in the receive FIFO becomes less than the value set in FCFTR register.
2. Manual PAUSE frame transmission
   PAUSE frame can also be transmitted by software instructions. When writing the timer value to MPR register, PAUSE frame transmission is started. With this method, PAUSE frame is transmitted only once.

### 21.4.20 PAUSE Frame Reception

If the PAUSE frame reception function is enabled, after a PAUSE frame is received, the next frame is not transmitted until the time indicated by the timer value of the PAUSE frame elapses. However, the on-going transmission of a frame continues when a PAUSE frame has been received.

### 21.4.21 Loopback Function

When the IBL bit in ECMR register is set to 1 and the full-duplex mode is enabled, the MII transmission data stream is looped back as MII reception data stream inside the MAC.

### 21.4.22 MAC Management Counters

Following MAC layer management counters are supported in ETNA:

- Transmit retry over counter: number of frames that were unable to be transmitted in 16 attempts
- PAUSE frame transmit counter: number of transmitted PAUSE frames
- Delayed collision detect counter: number of all delayed collisions after start of data transmission
- Lost carrier counter: number of times the carrier was lost during data transmission
- Frame transmit OK counter: number of frames successfully transmitted
- Frame transmit counter: number of frames successfully and erroneously transmitted
- Checksum error frame receive counter: number of received frames with a checksum error
- Frame receive error counter: number of receive error caused by RX_ER signal during reception
- Too short frame receive counter: number of received frames with a length of less than 64 bytes
- Too long frame receive counter: number of received frames with a length exceeding the value in RFLR register
- Residual bit frame receive counter: number of received frames containing residual bits
- Carrier extension lost counter: number of received frames with carrier extension lost
- Carrier extension error counter: number of received frames with an illegal carrier extension
- Multicast frame received counter: number of received multicast frames
- PAUSE frame receive counter: number of received PAUSE frames
- Frame receive OK counter: number of frames successfully received
- Frame receive counter: number of frames successfully and erroneously received

### 21.4.23 ETNA Operation

The ETNA consists of the following three function units:

- DMA transfer controller (E-DMAC):
  DMA transfer between the transmit/receive buffer in the memory and the transmit/receive FIFO
- MAC controller (MAC):
  Transmission/reception processing between the transmit/receive FIFO and the MII
- Transfer Switching Unit (TSU):
  CAM processing

Using its direct memory access (DMA) function, the E-DMAC performs DMA transfer of frame data between a user-specified Ethernet frame transmission/reception data storage destination (accessible memory space: transmit buffer/receive buffer) and the transmit/receive FIFO in the E-DMAC. The user cannot read and write data from and to the transmit/receive FIFO using the register.
To enable the E-DMAC to perform DMA transfer, information (data) including a transmit/receive data storage address and so forth, referred to as a descriptor, is required. The E-DMAC reads transmit data from the transmit buffer or writes receive data to the receive buffer according to the descriptor information. By arranging multiple descriptors as a descriptor row (list) (to be placed in a readable/writable memory space), multiple Ethernet frames can be transmitted or received continuously.
The E-DMAC consists of two systems: one for port 0 and the other for port 1, and both operate independently for transmission and reception.

The E-MAC constructs an Ethernet frame using the data written to the transmit FIFO and transmits the frame to the MII. It also performs a CRC check of an Ethernet frame received from the MII and deconstructs the frame to write to the receive FIFO.
The E-MAC consists of two controllers: E-MAC0 for port 0 and E-MAC1 for port 1, which correspond to E-DMAC0 and E-DMAC1, respectively.

The TSU, which is placed between the E-DMAC and MAC, references the CAM entry table to select one of the following tasks according to the Ethernet frame destination address (DA) input to the MAC.

- Receives data and writes to the receive FIFO.
- Discards data.

**Note:** **Figure 21.6, ETNA Data Path and Various Settings** shows the frame data path and an overview of each setting.

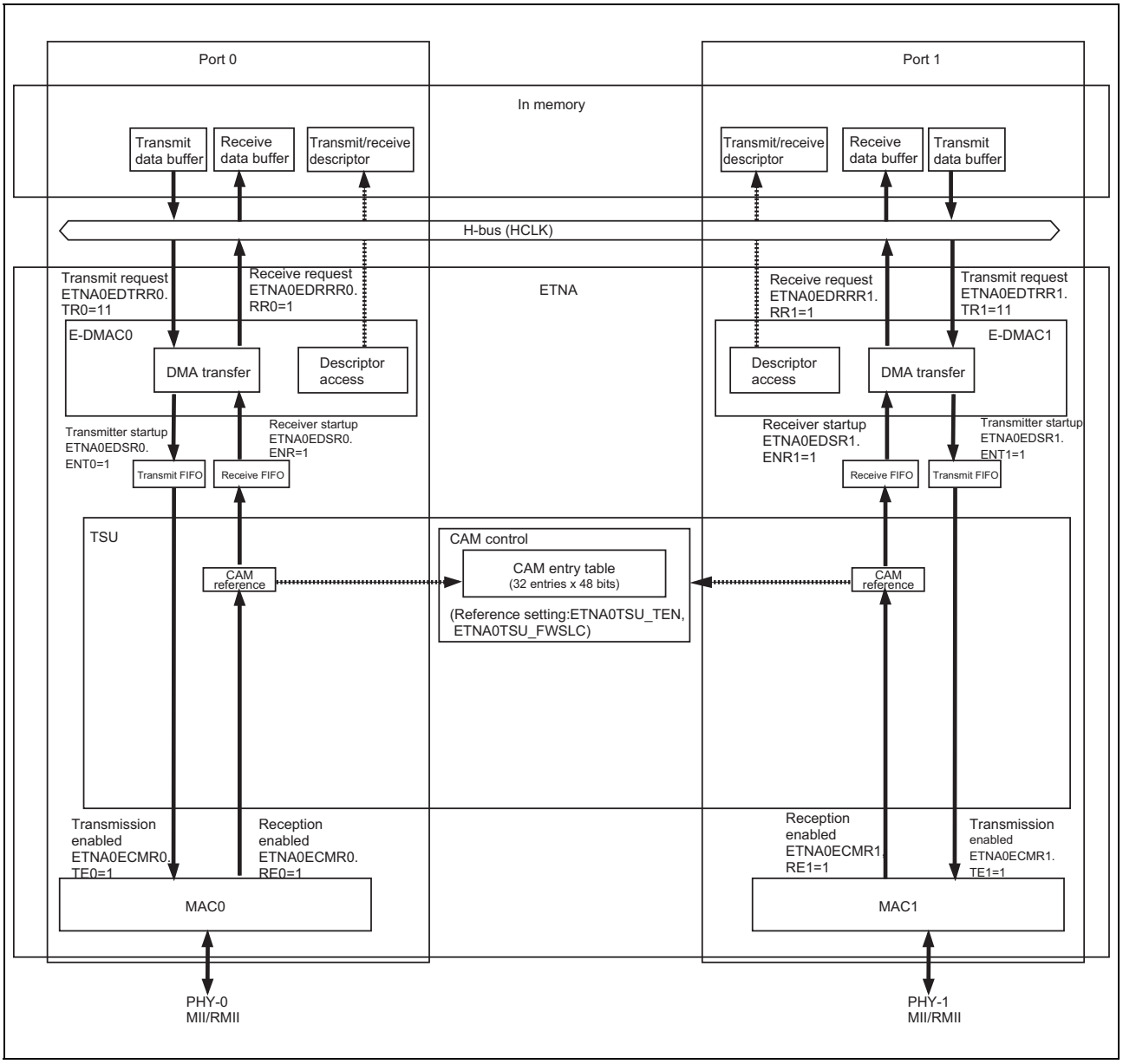



**Figure 21.6 ETNA Data Path and Various Settings**

### 21.4.24 Descriptors and Descriptor List

The E-DMAC performs DMA transfer according to the information (data), referred to as a descriptor, written in memory space. There are two types of descriptors: transmit descriptors and receive descriptors. Before a DMA transfer, DMA transfer information including a transmit/receive frame data storage address must be set by software.

The E-DMAC automatically starts reading a transmit/receive descriptor when the TRm[1:0] bits in ETNAnEDTRRm are set to 11 or the RRm bit in ETNAnEDRRRm is set to 1, and performs DMA transfer of frame data between the transmit/receive buffer and transmit/receive FIFO according to the information stored in the descriptor. After completion of Ethernet frame transmission/reception, the E-DMAC disables the descriptor valid/invalid bit and reflects the result of transmission/reception in the status bits.

Descriptors are placed in a readable/writable memory space. The address of the start descriptor (descriptor to be read first by the E-DMAC) is set in ETNAnTDLARm/ETNAnRDLARm. When multiple descriptors are prepared as a descriptor row (descriptor list), the descriptors are placed in continuous addresses (memory) according to the descriptor length set in the DLm0 and DLm1 bits in ETNAnEDMRm.

The E-DMAC consists of two systems: one for port 0 and the other for port 1. The DMAC for transmission and the DMAC for reception operate independently, and the DMAC for port 0 and the DMAC for port 1 operate independently. Place descriptors for transmission and reception and descriptors for port 0 and port 1 in those address spaces that do not overlap. If addresses are overlapped, E-DMAC does not successfully operate.

### 21.4.25 Transmit Descriptor

**Figure 21.7** shows the configuration of a transmit descriptor and the relationship with a transmit buffer.

The data of a transmit descriptor consists of TD0, TD1, TD2, and padding data in groups of 32 bits from top to end. The length of padding data is determined according to the descriptor length specified by the DLm0 and DLm1 bits in ETNAnEDMRm.

TD0 indicates whether the transmit descriptor is valid or invalid, and information about the descriptor configuration and status. TD1 indicates the length of data in a transmit buffer to be transferred (TDL) as specified by the descriptor. TD2 indicates the start address of a transmit buffer that holds data to be transferred (TBA).

Depending on the descriptor specification, one transmit descriptor can specify all transmit data of one frame (single-frame/single-buffer) or multiple descriptors can specify the transmit data of one frame (single-frame/multi-buffer). As an example of single-frame/multi-buffer operation, the data portion that is used in a fixed manner in each Ethernet frame transmission can be referenced by multiple descriptors. For example, multiple descriptors can share the destination address and transmit source address in an Ethernet frame, and the remaining data can be stored in each separate buffer.

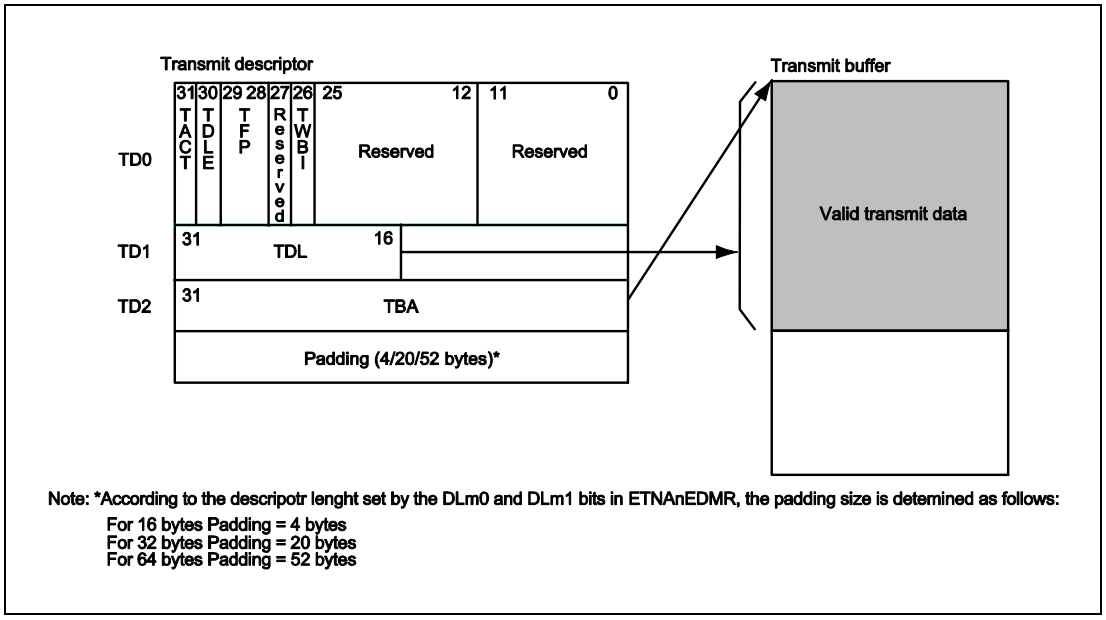



**Figure 21.7 Relationship between Transmit Descriptor and Transmit Buffer**

**Note:** Transmit descriptor ring has to be prepared at least five buffers. If transmit descriptor ring is set to less than five buffers, the descriptor ring may be looped two times by one transmit request. Because fetching all descriptors in the ring finishes before the write back process of the first buffers transmission completion.

### (1) Transmit Descriptor 0 (TD0)

Before the TRm[1:0] bits in ETNAnEDTRRm are set to 11, the user sets whether the bits of the descriptor are valid or invalid bit and sets other descriptor configuration. After Ethernet frame transmission, the E-DMAC disables the valid/invalid bits of the descriptor and writes status information. This operation is referred to as write-back.

When using TD0, the user should write desired values to bits 31 to 28 and 26 according to the descriptor configuration. Bits 27 and 25 to 0 should be cleared to 0.

**Table 21.81 TD0 description (1/2)**

| Bit | Bit name | Value after reset | R/W | Description |
|---|---|---|---|---|
| 31 | TACT | 0 | R/W | Transmit Descriptor Valid/Invalid<br>Indicates whether the corresponding descriptor is valid or invalid. To make this bit valid, store transmit data in a transmit buffer (user-specified transmit data storage destination) beforehand, then write 1 to this bit. The E-DMAC clears this bit to 0 after data transfer.<br>0: Indicates that this transmit descriptor is invalid<br>Indicates the initial setting state, the state after 0 is written, or (in case the user writes 1 to this bit) that this bit is cleared to 0 because the E-DMAC data transfer processing is completed.<br>If this state is recognized when the E-DMAC reads a descriptor, the E-DMAC clears the TRm[1:0] bits in ETNAnEDTRRm to 0, and halts transfer operation related to transmission by the E-DMAC.<br>1: Indicates that this transmit descriptor is valid<br>After the user writes 1 to this bit, this bit indicates that data is not transferred yet or data is being transferred.<br>When there is a descriptor row (descriptor list) consisting of multiple continuous descriptors, the E-DMAC can continue operation when this bit of the next descriptor is valid. |
| 30 | TDLE | 0 | R/W | Transmit Descriptor List End<br>Indicates whether the corresponding descriptor is the last descriptor of the descriptor row (descriptor list).<br>0: Not last descriptor<br>After transfer of the corresponding descriptor, the E-DMAC reads the next one in the list of continuous descriptors.<br>1: Last descriptor<br>After transfer of the corresponding descriptor, the E-DMAC reads the descriptor placed at the address indicated by ETNAnTDLARm. |

Table 21.81 TD0 description (2/2)

| Bit | Bit name | Value after reset | R/W | Description |
|---|---|---|---|---|
| 29, 28 | TFP[1:0] | 00 | R/W | Transmit Frame Position<br>These bits indicate whether information of this descriptor represents information about the start, middle, or end of the transmit frame.<br>00: The information of the descriptor represents information about the middle of the frame.<br>01: The information of the descriptor represents information about the end of the frame.<br>10: The information of the descriptor represents information about the start of the frame.<br>11: The information of the descriptor represents all information about the frame (single-frame/single-descriptor (single-buffer)).<br>Reference<br>When one frame is divided for use, the method of specifying this bit for a descriptor row according to the number of divisions is described below.<br>• For single-frame/single-descriptor operation<br>First descriptor: TFP[1:0] = 11<br>• For single-frame/two-descriptor operation<br>First descriptor: TFP[1:0] = 10<br>Second descriptor: TFP[1:0] = 01<br>• For single-frame/three-descriptor operation<br>First descriptor: TFP[1:0] = 10<br>Second descriptor: TFP[1:0] = 00<br>Third descriptor: TFP[1:0] = 01<br>When the number of divisions is large, a descriptor row is configured by adding intermediate descriptors with TFP[1:0] = 00. |
| 27 | — | All 0 | — | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 26 | TWBI | 0 | R/W | Write-Back Completion Interrupt Notification<br>0: Does not notify of a write-back completion interrupt<br>1: After a write-back operation to this descriptor is complete, this bit sets the TWBm and TWBm bits in ETNAnEESRm to 11 and notifies the CPU of a write-back completion interrupt.<br>This bit is valid only for the descriptor including the end of transmit frame (TFP = 01 or 11). This bit is cleared to 0 by the E-DMAC write-back operation. |
| 25 to 0 | — | All 0 | — | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (2) Transmit Descriptor 1 (TD1)

TD1 indicates the data length of the transmit buffer used by the corresponding descriptor.

The user should set TD1 before the start of a read by the E-DMAC.

Table 21.82 TD1 description

| Bit | Bit name | Value after reset | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | TDL | All 0 | R/W | Transmit Buffer Data Length (in bytes)<br>These bits indicate the data length of the corresponding transmit buffer in bytes. The maximum length is 64 Kbytes to 32 bytes ($FFE0_H$). |
| 15 to 0 | — | All 0 | — | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (3) Transmit Descriptor 2 (TD2)

TD2 indicates the start address of the corresponding 32-bit width transmit buffer. An address value should be specified in a longword boundary.

**Table 21.83 TD2 description**

| Bit | Bit name | Value after reset | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | TBA | All 0 | R/W | Transmit Buffer Start Address<br>These bits set the start address of the corresponding transmit buffer in a 4-byte boundary. |

If descriptors are set below, the E-DMAC does not return to normal operation until a system reset is performed.

- TFP (transmit frame position) is not logically correct
  Example: The TFP bits are set to 11 in a descriptor (descriptor A) and the TFP bits are set to 01 in the next descriptor (descriptor B). This specification means that there is no descriptor indicating the start of the transmit frame specified by descriptor B.
- TBL (transmit buffer length) is set to 0

When one transmit frame is divided into three parts or more with transmit descriptors, the E-DMAC performs the following write-back operation:

- A write-back operation is performed for a transmit descriptor including information for the start of the transmit frame (TFP = 10 or 11) and for a transmit descriptor including information for the end of the frame (TFP = 01 or 11).
- A write-back operation is not performed for a transmit descriptor for the middle of the frame (TFP = 00).

Before changing a transmit descriptor with the software, make sure that a write-back operation has been performed (TACT = 0) for the transmit descriptor including information for the end of the frame (TFP = 01 or 11) to avoid overwriting (re-setting) an unprocessed transmit descriptor.

### 21.4.26 Receive Descriptor

**Figure 21.8** shows the relationship between a receive descriptor and receive buffer.

The data of a receive descriptor consists of RD0, RD1, RD2, and padding data in groups of 32 bits from top to end. The length of padding data is determined according to the descriptor length specified by the DLm0 and DLm1 bits in ETNAnEDMRm.

RD0 indicates whether the receive descriptor is valid or invalid, and information about descriptor configuration and status. RD1 indicates the length of data that can be received in the receive buffer specified by the descriptor (RBL) and the length of the received frame data (RDL). RD2 indicates the start address of the receive buffer for storing receive data (RBA).

Depending on the descriptor specification, one receive descriptor can specify the storing of all receive data of one frame in a receive buffer (single-frame/single-buffer) or multiple descriptors can specify the storing of the receive data of one frame in receive buffers (single-frame/multi-buffer). As an example of single-frame/multi-buffer operation, suppose that a row of multiple descriptors (descriptor list) is prepared, RBL of each descriptor is 480 bytes, and a 1514-byte Ethernet frame is received. In such a case, the received Ethernet frame is transferred sequentially to buffers, 480 bytes for each buffer, starting with the first descriptor. Only the last 74 bytes are transferred to the fourth buffer. When a frame longer than RBL of a descriptor is received, the E-DMAC transfers the remaining data to the receive buffer by using the subsequent descriptors. As an example of efficient single-frame/multi-buffer operation, information items on different processing layers in an Ethernet frame can be separated from each other by using different buffers. For example, the destination address, transmit source address, type field data in an Ethernet frame, and 18 bytes padding can be stored in buffer 1 (set RBL to 32 bytes) and the remaining data can be stored in buffer 2 (set RBL to 1504 bytes). All receive frames, of course, can be stored in a single buffer if multiple descriptors are prepared and RBL of each descriptor is set to more than 1514 bytes (maximum Ethernet frame length).

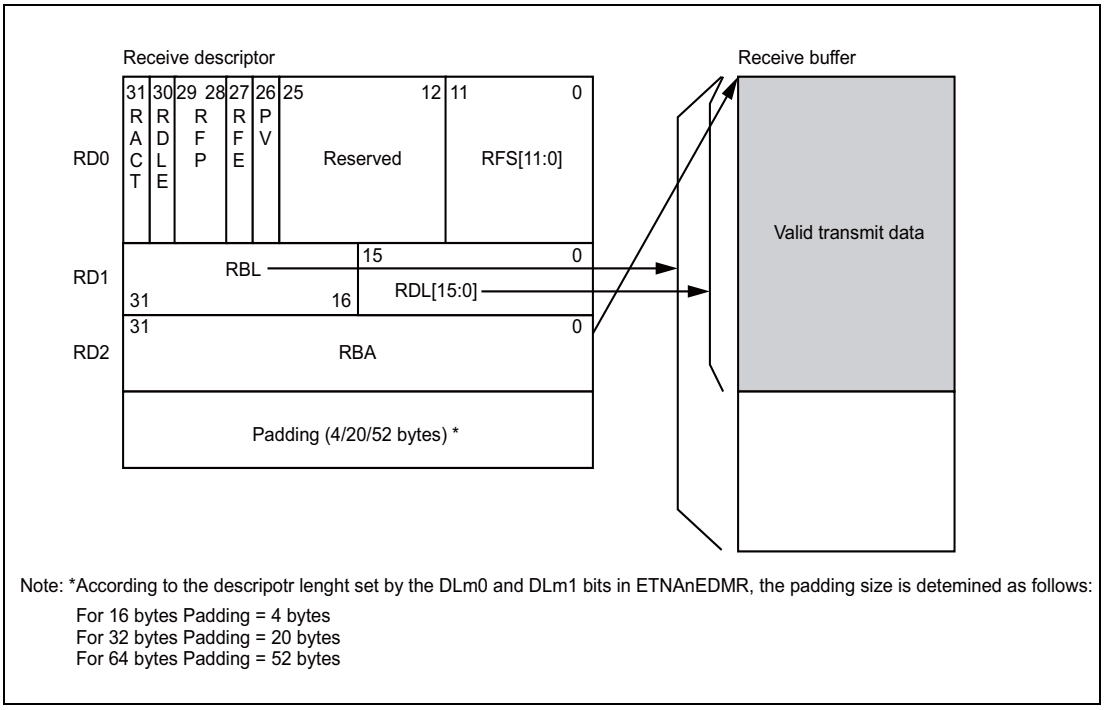



**Figure 21.8 Relationship between Receive Descriptor and Receive Buffer**

### (1) Receive Descriptor 0 (RD0)

The user sets whether the bits of the descriptor are valid or invalid and whether the descriptor represents the end of the descriptor list in RD0 before the RRm bit in ETNAnEDRRRm is set to 1 and the start of a read by the E-DMAC. After receive DMA transfer of an Ethernet frame by the E-DMAC, the E-DMAC disables the valid/invalid bits of the descriptor and writes status information. This operation is referred to as write-back.

When using RD0, the user should write desired values to bits 31 and 30 according to the descriptor configuration. Bits 29 to 0 should be cleared to 0.

**Table 21.84 RD0 description (1/3)**

| Bit | Bit name | Value after reset | R/W | Description |
|---|---|---|---|---|
| 31 | RACT | 0 | R/W | Receive Descriptor Valid/Invalid<br>Indicates whether this descriptor is valid or invalid. To make this bit valid, prepare a receive buffer (user-specified receive data storage destination) beforehand, then write 1 to this bit. The E-DMAC clears this bit to 0 after data transfer or when a receive error has occured.<br>0: Indicates that this receive descriptor is invalid<br>Indicates the initial setting state, the state after 0 is written to, or (in case the user writes 1 to this bit) that this bit is cleared to 0 because the E-DMAC data transfer processing is completed<br>If this state is recognized when the E-DMAC reads a descriptor, the E-DMAC clears the RRm bit in ETNAnEDRRRm to 0, and halts transfer operation related to reception by the E-DMAC<br>1: Indicates that this receive descriptor is valid<br>Indicates that data is not transferred yet after the user writes 1 to this bit, or that data is being transferred<br>When there is a descriptor row (descriptor list) consisting of multiple continuous descriptors, the E-DMAC can continue operation when this bit of the next descriptor is valid |
| 30 | RDLE | 0 | R/W | Receive Descriptor List End<br>Indicates whether this descriptor is the last descriptor of the descriptor row (descriptor list).<br>0: Not last descriptor<br>After transfer of this descriptor, the E-DMAC reads the next one in the list of continuous descriptors<br>1: Last descriptor<br>After transfer of this descriptor, the E-DMAC reads the descriptor placed at the address indicated by ETNAnRDLARm |

Table 21.84 RD0 description (2/3)

| Bit | Bit name | Value after reset | R/W | Description |
|---|---|---|---|---|
| 29, 28 | RFP[1:0] | 00 | R/W | Receive Frame Position 1, 0<br>The E-DMAC indicates by write-back operation whether information of the corresponding descriptor represents information about the start, middle, or end of the receive frame.<br>00: The information of the descriptor represents information about the middle of the frame<br>01: The information of the descriptor represents information about the end of the frame<br>10: The information of the descriptor represents information about the start of the frame<br>11: The information of the descriptor represents all information about the frame (single-frame/single-descriptor (single-buffer))<br>Reference<br>The relationship between a frame after reception of one frame and a descriptor is described below.<br>• For single-frame/single-descriptor operation<br>First descriptor: RFP[1:0] = 11<br>• For single-frame/two-descriptor operation<br>First descriptor: RFP[1:0] = 10<br>Second descriptor: RFP[1:0] = 01<br>• For single-frame/three-descriptor operation<br>First descriptor: RFP[1:0] = 10<br>Second descriptor: RFP[1:0] = 00<br>Third descriptor: RFP[1:0] = 01<br>When the number of divisions is large, a descriptor row is configured by adding intermediate descriptors with RFP[1:0] = 00. |
| 27 | RFE | 0 | R/W | Receive Frame Error Occurrence<br>Indicates that an error occurred in the receive frame.<br>0: RFS11 to RFS0 are all 0<br>1: One of RFS11 to RFS0 is 1<br>Each of RFS8 to RFS0 can be masked by using ETNAnTRSCERm. RFS11 to RFS9 cannot be masked.<br>This bit is set by the E-DMAC write-back operation. |
| 26 | PV | 0 | R/W | Padding Insertion<br>Indicates whether the padding specified by ETNAnRPADIRm was inserted in the receive frame processed with this descriptor or not.<br>0: No padding inserted<br>1: Padding inserted<br>This bit can be changed by the E-DMAC write-back processing. |
| 25 to 12 | — | All 0 | — | Reserved<br>These bits are always read as 0. The write value should always be 0. |

Table 21.84 RD0 description (3/3)

| Bit | Bit name | Value after reset | R/W | Description |
|---|---|---|---|---|
| 11 to 0 | RFS[11:0] | All 0 | R/W | Receive Frame Status<br>Indicate the status of the corresponding frame. A bit below, when set to 1, indicates the occurrence of the corresponding event. If an event indicated by any of RFS9 to RFS0 occurs, the frame is not received completely.<br>RFS[11:10]: Reserved<br>RFS[9]: Receive FIFO overflow (corresponding to the RFOF bit in ETNAnEESRm)<br>RFS[8]: Detection of reception abort (Corresponding to the RABT bit in ETNAnEESRm)<br>RFS[7]: Multicast address frame received (corresponding to the RMAF bit in ETNAnEESRm)<br>RFS[6]: Reserved<br>RFS[5]: Reserved<br>RFS[4]: Residual-bit frame receive error (corresponding to the RRF bit in ETNAnEESRm)<br>RFS[3]: Long frame receive error (corresponding to the RTLF bit in ETNAnEESRm)<br>RFS[2]: Short frame receive error (corresponding to the RTSF bit in ETNAnEESRm)<br>RFS[1]: PHY-LSI receive error (corresponding to the PRE bit in ETNAnEESRm)<br>RFS[0]: CRC error on receive frame (corresponding to the CERF bit in ETNAnEESRm) |

### (2) Receive Descriptor 1 (RD1)

In RD1, the user specifies the data length of a receive buffer usable by the corresponding descriptor. After reception of a frame, RD1 indicates the length of a frame received by the E-DMAC.

The user should set RD1 before the start of a read by the E-DMAC.

**Table 21.85 RD1 description**

| Bit | Bit name | Value after reset | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | RBL | All 0 | R/W | Receive Buffer Data Length (in bytes, to be specified with a 32-byte boundary)<br>These bits set the length of data that can be received by the corresponding receive buffer with an integral multiple of 32 bytes.<br>The maximum receive buffer data length is<br><br>64 Kbytes – 32 bytes ($FFE0_H$). |
| 15 to 0 | RDL | All 0 | R | Receive Data Length<br>These bits indicate the data length of a receive frame stored in the receive buffer.<br>Receive data transferred to the receive buffer does not include CRC data (4 bytes) placed at the end of a frame. As a receive frame length, the number of bytes (valid data bytes) not including CRC data are reported.<br>In single-frame/multi-buffer (descriptor) operation, only the receive data length of the last descriptor is valid. The receive data length of an intermediate descriptor has no meaning.<br>The padding size set by ETNAnRPADIRm is not included. |

### (3) Receive Descriptor 2 (RD2)

RD2 indicates the start address of the corresponding receive buffer. Set the start address of a receive buffer with a 32-byte boundary.

**Table 21.86 RD1 description**

| Bit | Bit name | Value after reset | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | RBA | All 0 | R/W | Receive Buffer Start Address<br>These bits set the start address of the corresponding receive buffer with a 32-byte boundary. |

The E-DMAC performs DMA transfer for a receive frame from the address specified by RBA (receive buffer address) to the receive buffer in 32-byte units. RBL (receive buffer length) must be set to be an integral multiple of 32 bytes.

If data to be transferred is less than 32 bytes, invalid data will be written to memory.

[Example]

When the receive frame length is 170 bytes and the required receive buffer capacity is 192 bytes (32 bytes × 6), the sixth DMA-transfer causes invalid data to be written to the receive buffer (In the 32-byte DMA data, the former 10 bytes are valid and the latter 22 bytes are invalid).

Padding of the value 0 can be inserted into only one position in the receive frame by setting ETNAnRPADIRm. The padding size can be selected from 1 byte to 31 bytes in byte units. When padding is inserted into a receive frame, a receive buffer area equal to the total of “receive frame length and padding size”is required. ETNAnRPADIRm setting is valid for all receive frames.

RFE (receive frame error occurrence), PV (padding insertion), RFS (receive frame status) and RFL (receive frame length) are only set in the receive descriptor including information for the end of the frame (RFP = 01 or 11) by a write-back operation.

Before re-setting a receive descriptor with the software, completion of a write-back operation for the receive descriptor (RACT = 0) must be confirmed to avoid rewriting to (and re-setting) an unprocessed receive descriptor.

## 21.4.27 Descriptor and Transmit/Receive Buffer

### (1) Transmission

Each transmit descriptor specifies one transmit buffer. The E-DMAC transfers a transmit frame stored in a transmit buffer specified by a transmit descriptor to the transmit FIFO. Multiple transmit frames stored in transmit buffers specified by multiple descriptors can be connected into one transmit frame and transferred to the transmit FIFO.

**Figure 21.9** shows the relationship between the transmit descriptors and transmit buffers.

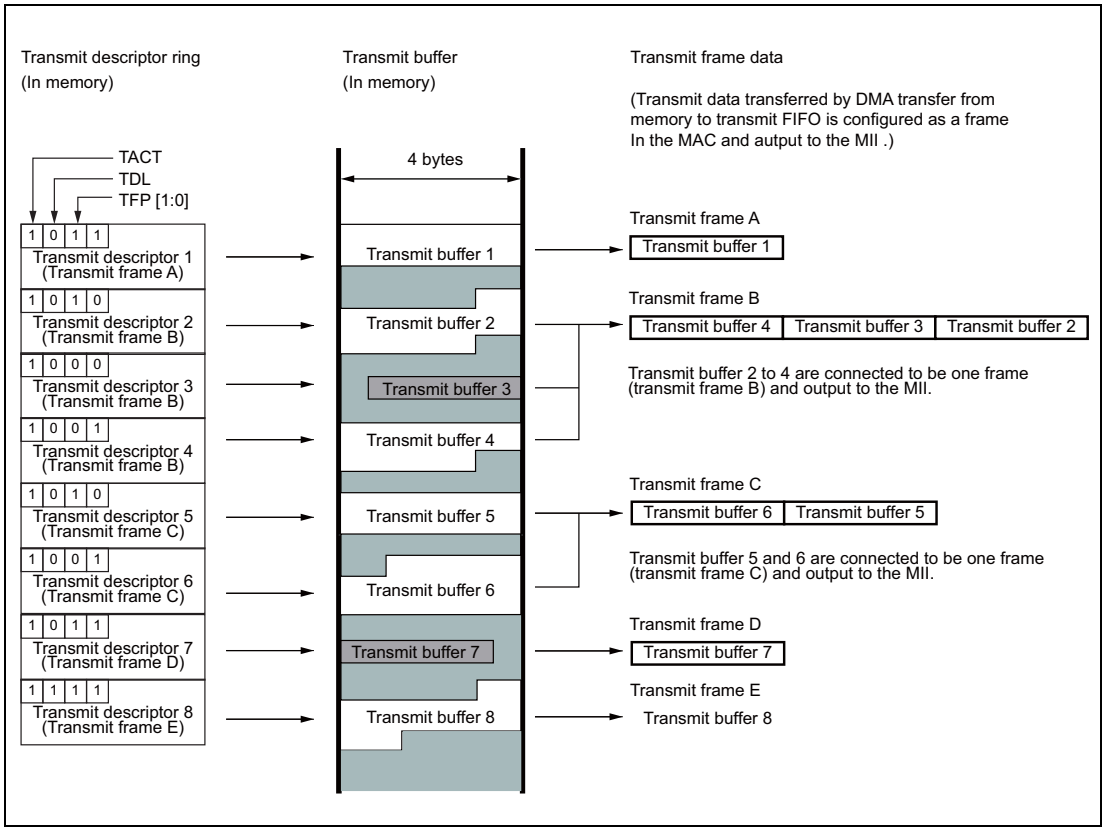



**Figure 21.9 Relationship between Transmit Descriptor and Transmit Buffer**

**Note:** Transmit descriptor ring has to be prepared at least five buffers. If transmit descriptor ring is set to less than five buffers, the descriptor ring may be looped two times by one transmit request. Because fetching all descriptors in the ring finishes before the write back process of the first buffers transmission completion

### (2) Reception

Each receive descriptor specifies one receive buffer. The E-DMAC receives a receive frame from the receive FIFO and stores it in a receive buffer specified by a receive descriptor. If the receive frame size exceeds the receive buffer size, the remaining data of the receive frame can be stored in a different receive buffer specified by a different receive descriptor. Thus, one receive frame can be stored in multiple receive buffers.

**Figure 21.10, Relationship between Receive Descriptor and Receive Buffer** shows the relationship between the receive descriptors and receive buffers.

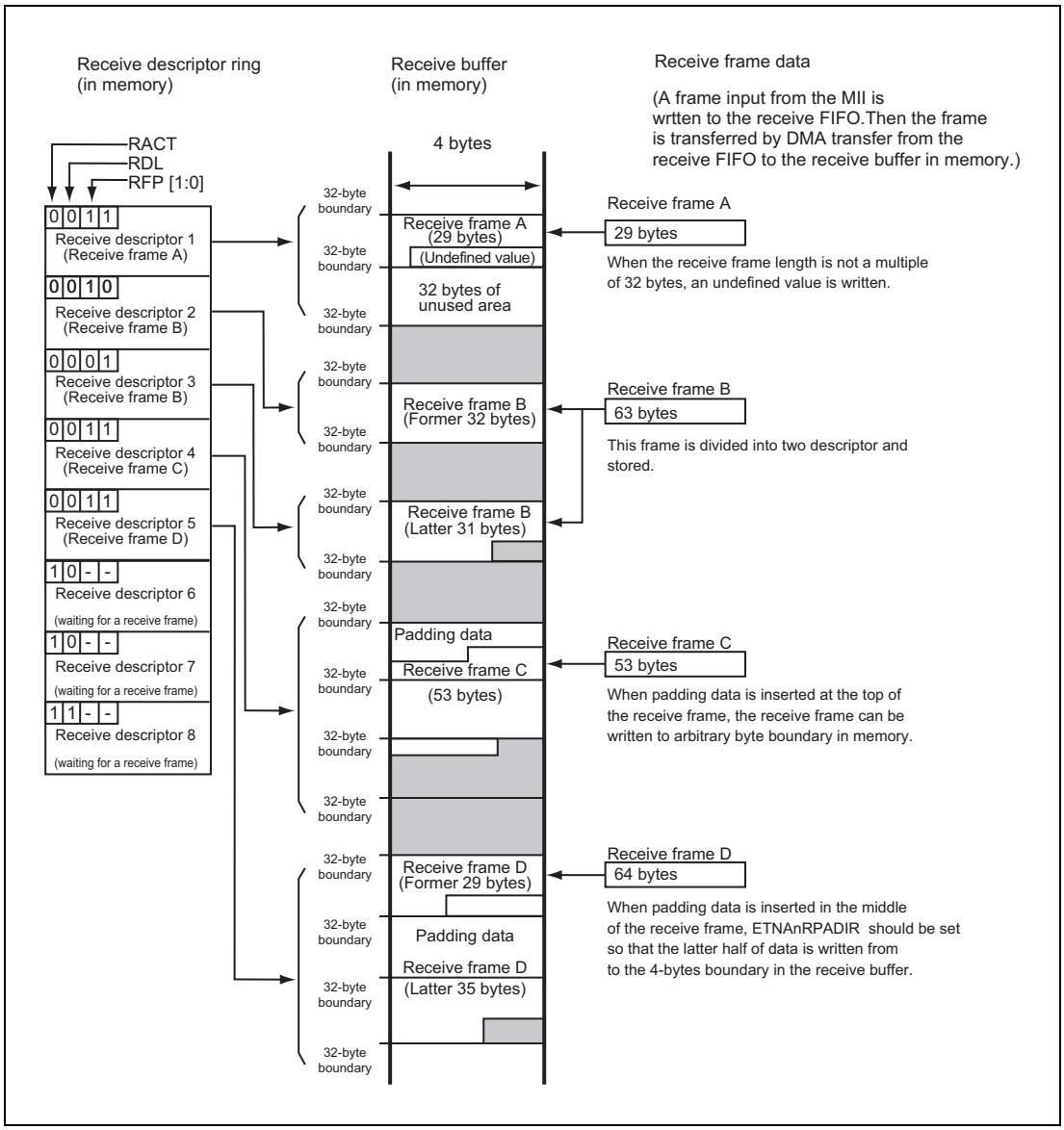



**Figure 21.10 Relationship between Receive Descriptor and Receive Buffer**

### 21.4.28 Descriptor Pointer

The E-DMACm controls the transmit and receive descriptor addresses in memory and the processing priority by using the following registers.

(1) Registers related to a transmit descriptor

- ETNAnTDLARm: Address of the start descriptor in a list of transmit descriptors.
- ETNAnTDFARm: Address of the transmit descriptor to be processed
- ETNAnTDFXRm: Address of the transmit descriptor that finished processing (set by a write-back operation) last
- ETNAnTDFFRm: Indicates whether the transmit descriptor for which the E-DMACm has just completed the write-back processing and whose start address is stored in ETNAnTDFXRm is at the end of the transmit descriptor queue (descriptor list).

(2) Registers related to receive descriptor:

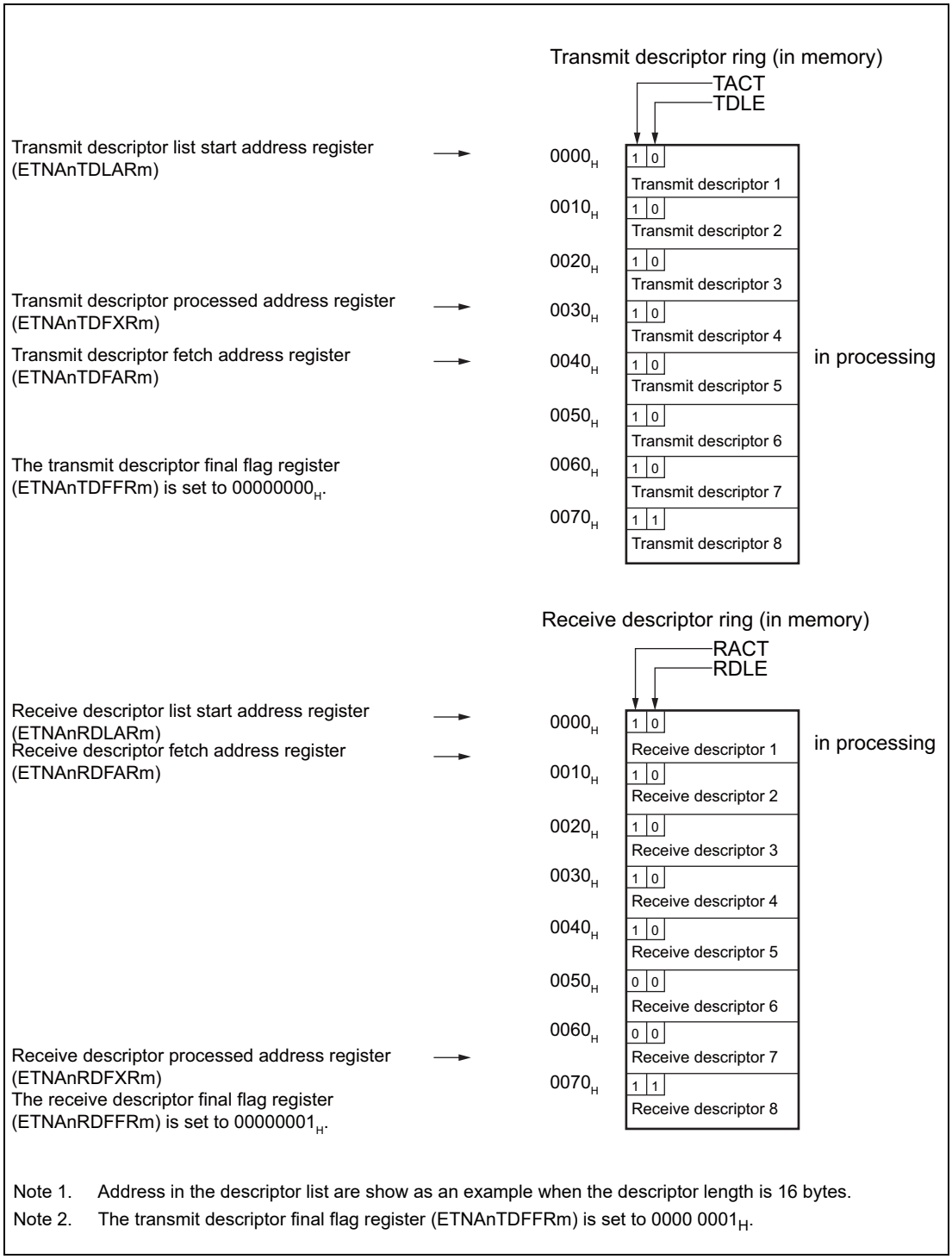
- ETNAnRDLARm: Address of the start descriptor in a list of receive descriptors.
- ETNAnRDFARm: Address of the receive descriptor to be processed
- ETNAnRDFXRm: Address of the receive descriptor that finished processing (set by a write-back operation) last
- ETNAnRDFFRm: Indicates whether the receive descriptor for which the E-DMACm has just completed the write-back processing and whose start address is stored in ETNAnRDFXRm is at the end of the receive descriptor queue (descriptor list).

Transmit descriptors and receive descriptors have a ring structure. When the TDLE (RDLE) value of the processed transmit (receive) descriptor is 0, the next descriptor will be processed. The next descriptor is the transmit (receive) descriptor at the address obtained by adding the processed transmit (receive) descriptor address to the descriptor length specified by the DLm[1:0] bit in ETNAnEDMRm. When the TDLE (RDLE) value of the processed transmit (receive) descriptor is 1, the transmit descriptor indicated by ETNAnTDLARm (ETNAnRDLARm) will be processed next. **Figure 21.11** shows the relationship between the transmit/receive descriptor ring and read pointer.

The transmit descriptor list must be large enough to point to five or more transmit frames. If four or less transmit frames are pointed to in a list, E-DMACm operation is not guaranteed. Accordingly, do not set that all the transmit descriptors in a ring are used by four or less descriptors. The receive descriptor list does not have this restriction. For example, one receive frame can use all receive descriptors in a list.

In the initial setting, the start address of a descriptor list must be set to ETNAnTDLARm (ETNAnRDLARm) and ETNAnTDFARm (ETNAnRDFARm), and the end descriptor address of the descriptor list to ETNAnTDFXRm (ETNAnRDFXRm) by the software.

The E-DMAC updates ETNAnTDFARm (ETNAnRDFARm), ETNAnTDFXRm (ETNAnRDFXRm) and the DLm[1:0] bit in ETNAnTDFFRm (DLm[1:0] bit in ETNAnRDFFRm) each time a descriptor is processed.




Note 1. Address in the descriptor list are show as an example when the descriptor length is 16 bytes.

Note 2. The transmit descriptor final flag register (ETNAnTDFFRm) is set to 0000 0001H.

**Figure 21.11 Relationship between Transmit/Receive Descriptor and Descriptor Pointing Registers**

### 21.4.29 Transmission Procedure and Processing Flow

When 11 is written to the TRm[1:0] bits in ETNAnEDTRRm with the TEm bit in ETNAnECMRm set to 1 and there is empty space of 32 bytes or more in the transmit FIFO, the E-DMAC reads the descriptor following the previously used descriptor from the transmit descriptor list (or the descriptor indicated by ETNAnTDLARm at the initial startup).

If the TACT bit of the read descriptor is set to 1 (valid), the E-DMAC sequentially reads transmit frame data from the transmit buffer start address specified by TD2 and transfers the data to the transmit FIFO. The E-DMAC configures a transmit frame and starts transmission to the MII. After DMA transfer of data equivalent to the buffer length specified in the descriptor, the following processing is carried out according to the TFP value.

- TFP = 10 (start of a frame)
  Descriptor write-back (writing 0 to the TACT bit) is performed after completion of DMA transfer.
- TFP = 01 or 11 (end of a frame)
  Descriptor write-back (writing 0 to the TACT bit and writing status) is performed after completion of frame transmission.
- TFP = 00 (frame continued)
  Descriptor write-back is not performed. The TACT bit retains the value 1.

As long as the TACT bit of a read descriptor is set to 1 (valid), the reading of E-DMAC descriptors and the transmission of frames continue.

When a descriptor with the TACT bit cleared to 0 (invalid) is read, the E-DMAC performs the following processing and completes transmit processing.

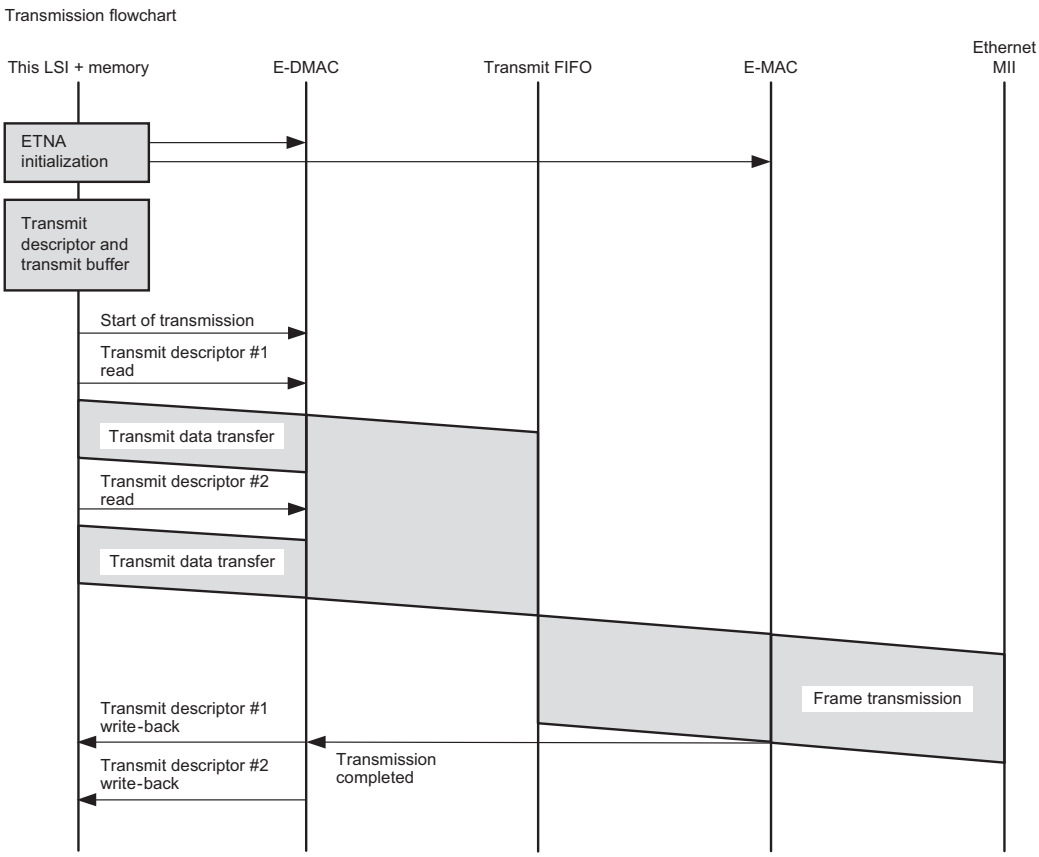
- Clears the TRm[1:0] bits in ETNAnEDTRRm to 00.
- Writes the TCm bits in ETNAnEESRm to 11 and generates an interrupt.

The E-DMAC can store up to four frames of data in the transmit FIFO.

When one of the following conditions is met, the MAC transmit processing unit reads the data from the transmit FIFO to configure a frame and sends it to the MII.

- One or more frame of data is stored in the transmit FIFO.
- Transmit FIFO full.

**Figure 21.12** shows an example of transmission flow.




[Legend]

ETNA initialization: Executes a software reset with the SWR bit in EDMR set to 1.
Transmit descriptor and transmit setting: Sets transmit descriptors and transmit buffers, and sets E-MAC and E-DMAC registers, then writes 11 to the TE bit in ECMR and the TR bit in EDTRR.
Start of transmission: Occurs when 1 is written to the TE bit in ECMR and 11 is written to the TR bit in EDTRR.
Transmit descriptor read: The E-DMAC reads a transmit descriptor.
Transmit data transfer: Writes transmit data to the transmit FIFO by using DMA transfer by the E-DMAC.
Transmit descriptor write-back: The E-DMAC writes 0 to the TACT bit.

**Figure 21.12 Sample Transmission Flowchart (Single-Frame/Two-Description in store and forward mode)**

**Figure 21.13** shows the status change of the MAC transmitter. This operation is common among port 0 and port 1.

1. When the TEn bit in ETNAnECMRm is set, the transmitter enters the transmit idle state.
2. When a transmit request is issued by the transmit E-DMAC, the MAC sends the preamble to MII after a transmission delay caused by the carrier detection and frame interval time. If full-duplex transfer is selected, which does not require carrier detection, the preamble is sent as soon as a transmit request is issued by the E-DMAC.
3. The transmitter sends the SFD, data, and CRC sequentially. At the end of transmission, the transmit E-DMAC generates a transmission complete interrupt (TC). If a collision or the carrier-not-detected state occurs during data transmission, these are reported as interrupt sources.
4. After waiting for the frame interval time, the transmitter enters the idle state, and if there is more transmit data, continues transmitting.

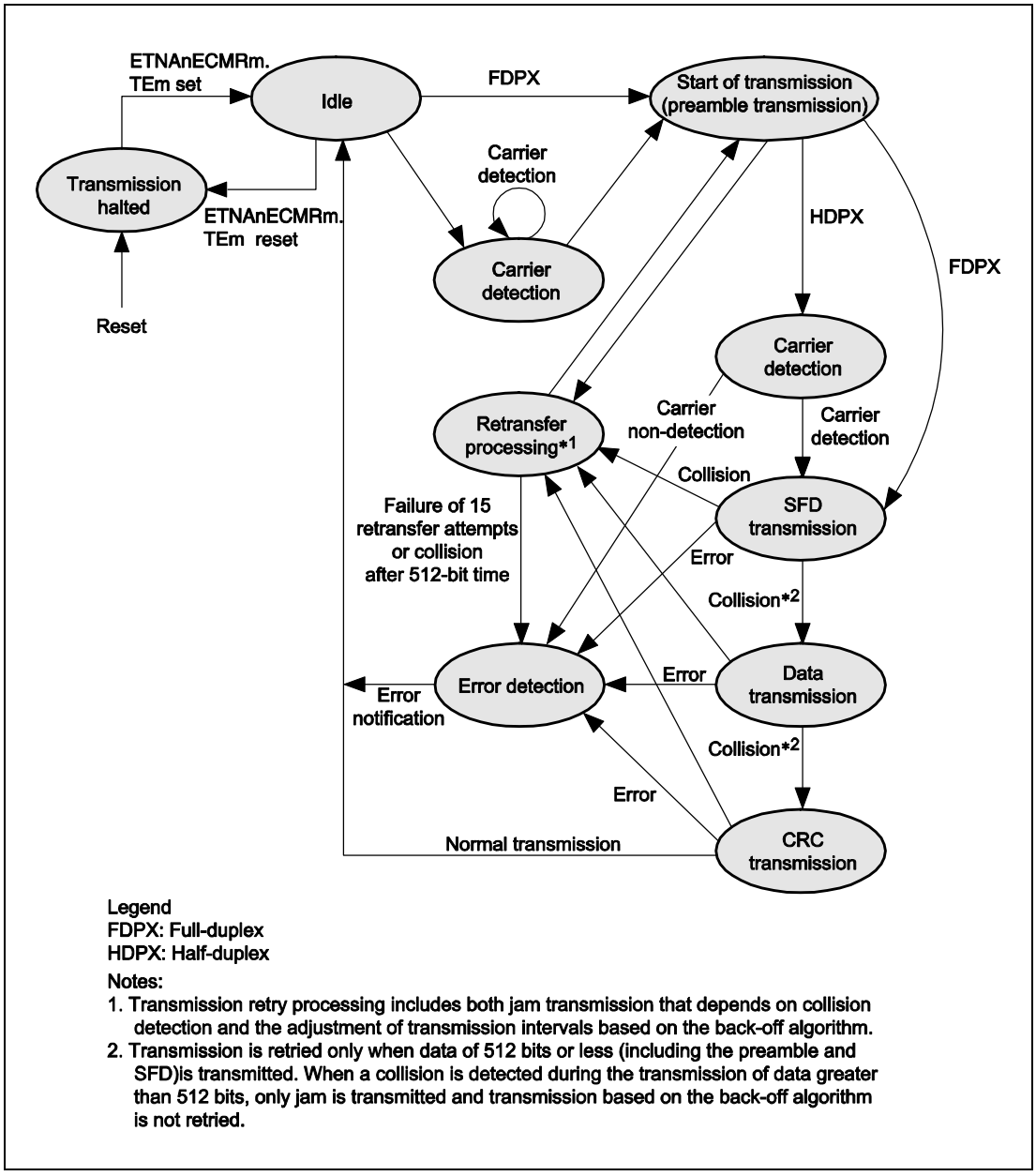



Figure 21.13 MAC Transmitter State Transitions

## 21.4.30 Transmission Error Processing

### (1) Transmit Descriptor Empty

When the TFP bits of the descriptor previously processed are set to 00 or 10 and the TACT bit of the read transmit descriptor is set to 0 (invalid), a transmit descriptor empty state is determined and 1 is written to the TDE bit in ETNAnEESRm, and then an interrupt is issued.

When a transmit descriptor state is empty, start transmission processing after a software reset.

## 21.4.31 Reception Procedure and Processing Flow

The MAC receiver separates the frame from the MII into preamble, SFD, data and CRC, and transfers the fields from DA (destination address) to the data to the receive FIFO. Up to 24 frames can be written in the receive FIFO. **Figure 21.14** shows the status change of the MAC receiver. This operation is common among port 0 and port 1.

1. When the REm bit in ETNAnECMRm is set to 1, the receiver enters the receive idle state.
2. When an SFD (start frame delimiter) is detected after a receive packet preamble, the receiver starts receive processing. A frame with an invalid pattern is discarded.
3. In normal mode, the receiver starts data reception if the destination address of the frame specifies the one that the product has or broadcast or multicast. In promiscuous mode, data reception starts regardless of the frame type.
4. Following data reception from the MII, the receiver carries out a CRC check. The result is indicated as a status bit in the descriptor after the frame data has been written to the receive FIFO. Reports an error status in the case of an abnormality.

After one frame has been received, if the REm bit in ETNAnECMRm is set to 1, the receiver prepares to receive the next frame.

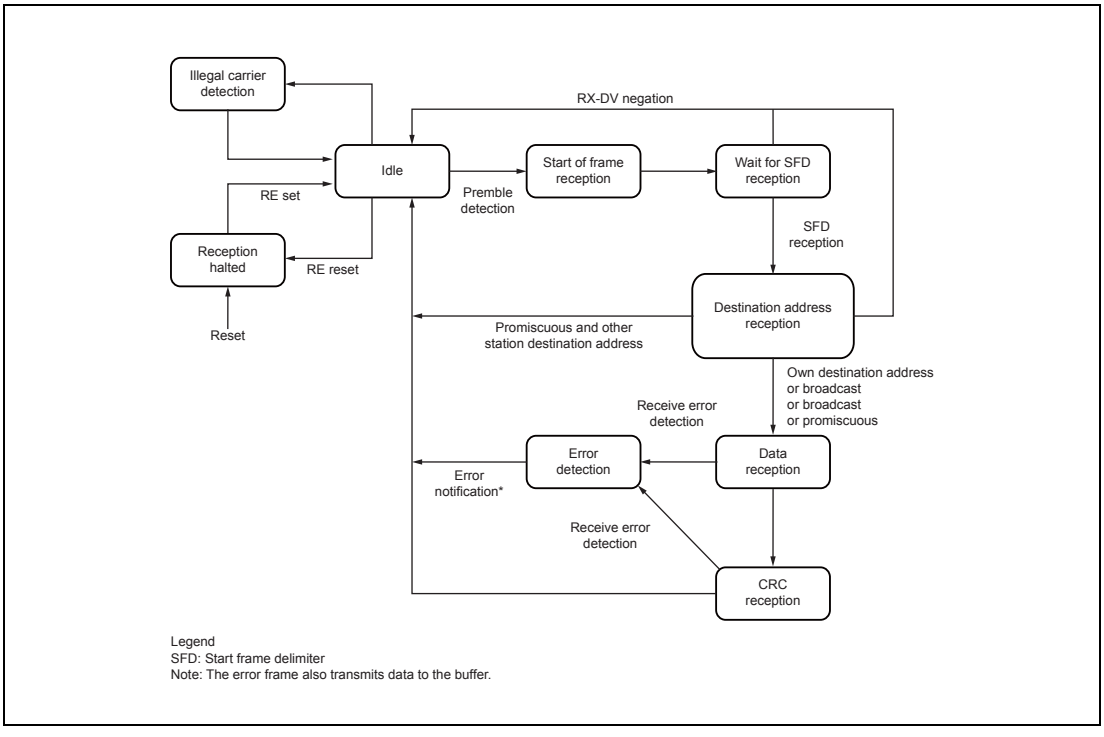



**Figure 21.14 MAC Receiver State Transitions**

CAM evaluation can be referenced during frame processing in reception (for details on the CAM function, see **Section 21.4.33, CAM Function**).

When 1 is written to the RRm bit in ETNAnEDRRRm while the REm bit in ETNAnECMRm is set to 1, the E-DMAC reads the descriptor following the previously used descriptor from the receive descriptor list (or the descriptor indicated by ETNAnRDLARm at the initial startup) then enters the receive wait state. If 32 bytes or more of data or the last byte of the receive frame is stored in the receive FIFO, the E-DMAC transfers receive FIFO data to the receive buffer specified by RD2 according to the receive descriptor with the RACT bit set to 1 (valid).

If the data length of a received frame is longer than the buffer length specified by RD1, the E-DMAC performs a write-back operation to the descriptor (set RFP to 10 or 00) when the buffer is full, then reads the next descriptor. The E-DMAC then continues to transfer data to the receive buffer specified by the new RD2.

When the following conditions are satisfied, a write-back operation is performed for the descriptor (RFP = 11 or 01), 11 is written to the FR bits in ETNAnEESRm, and an interrupt is issued.

- The receive buffer has been full during DMA transfer.
- DMA transfer to the receive buffer of the last byte of the receive frame has been completed.

After the reception processing of the frame, the next descriptor reading standby state begins. At this time, if 32 bytes or more of data or the last byte of the receive frame is stored in the receive FIFO, the next receive descriptor process is performed continuously.

When the RACT bit of the read receive descriptor is 0 (invalid), the receive descriptor empty state is determined and the RDE bit in ETNAnEESRm is written to 1, and then an interrupt is issued.

To receive frames continuously, set the RNC bit in ETNAnRMCRm to 1. The value after reset is 0.

**Figure 21.15** shows an example of reception flow.

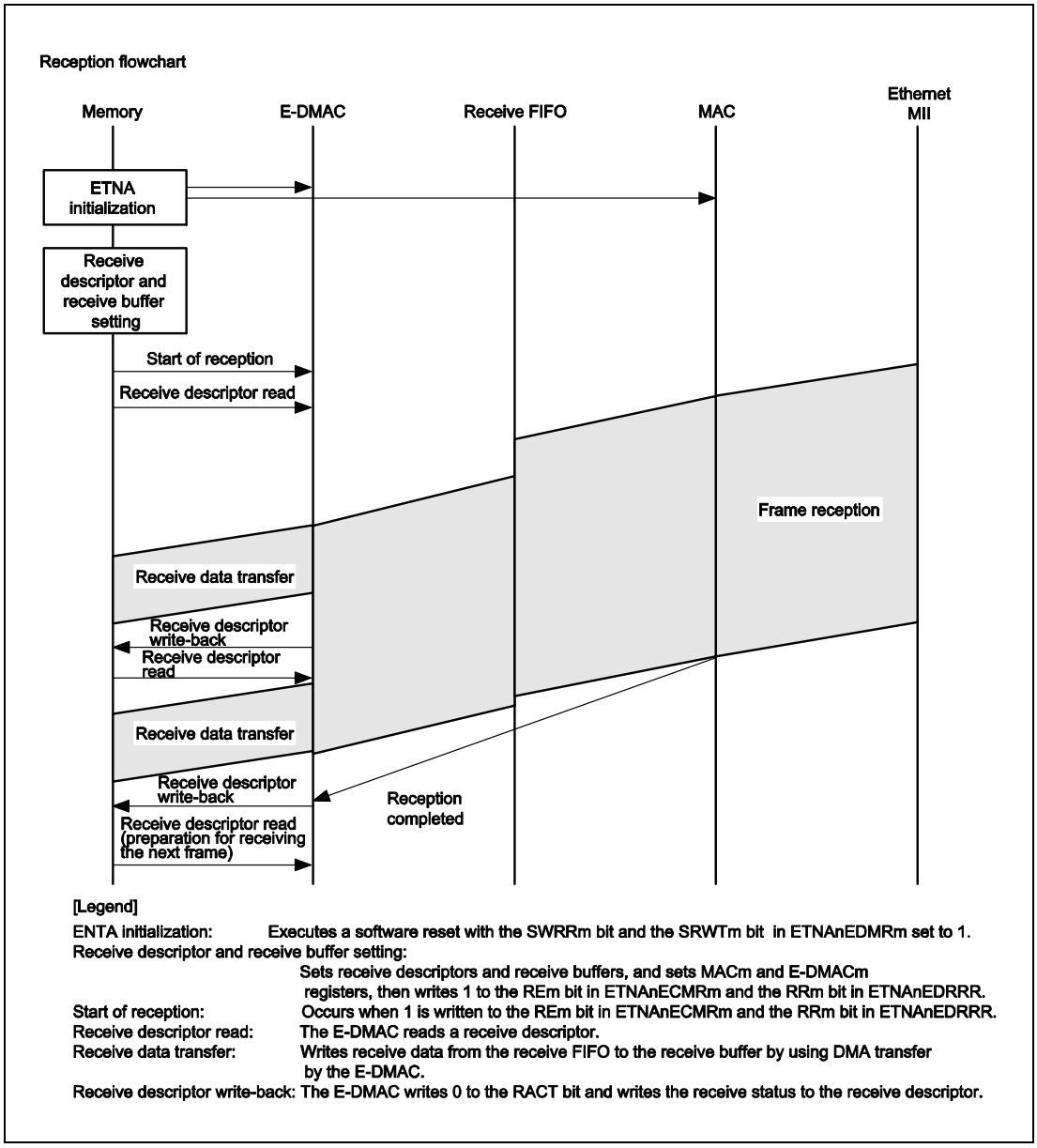



**Figure 21.15 Sample Reception Flowchart (Single-Frame/Two-Descriptor)**

## 21.4.32 Reception Error Processing

### (1) Reception Error

When a reception error occurs, the FR and RABT bits in ETNAnEESRm are set to 1 and an interrupt is issued after a write-back operation for the receive descriptor related to the reception error frame.

If a reception error occurs when the length of the frame received from the MII is less than 32 bytes, DMA transfer to the receive buffer for the frame is not performed. At this time, the receive frame is discarded in the E-DMAC (flush function). However, if padding is inserted in the receive frame by ETNAnRPADIRm, the flush function is performed when the frame length including the padding bytes is less than 32 bytes.

### (2) Receive FIFO Overflow

In any of the following cases, the MAC cannot receive frames from the MII because it has no space to store receive frames, and all the receive frames that have been transferred to the MAC will be discarded in the MAC (receive FIFO overflow).

- Receive FIFO is full of data waiting for DMA transfer (the receive FIFO has no space).
- The number of receive frames waiting for DMA transfer is 24 in total (the receive frame information managing area has no empty space; up to 24 frames can be managed).

If an overflow occurs due to the former case, the RFE bit in ETNAnEESRm is set to 1 and an interrupt is generated. If an overflow occurs due to the latter case, the RFCOF bit in ETNAnEESRm is set to 1 and an interrupt is generated. Each time a receive frame is discarded due to an overflow, ETNAnRMFCRm is incremented. However, ETNAnRMFCRm is not incremented for a receive frame that is cut off due to insufficient receive FIFO space. If a receive frame is cut off due to insufficient receive FIFO space (the frame is partially stored in the receive FIFO), the E-DMAC performs the following operation:

- Performs DMA transfers for the cut-off frame stored in the receive FIFO to the receive buffer.
- After the DMA transfer, performs a write-back operation on the receive descriptor.
- After the write-back operation, sets the ROC bit in ETNAnEESRm to 1 and generates an interrupt.

When the receive FIFO is full of data waiting for DMA transfer, frame reception from the MII can be resumed if DMA transfer is performed from the receive FIFO to the receive buffer and 32 bytes or more of empty space is generated in the receive FIFO. When the number of receive frames waiting for DMA transfer is 24 in total, frame reception from the MII can be resumed if one or more frame has been DMA transferred from the receive FIFO to the receive buffer. For restarting frame reception from the MII, when the E-DMAC resumes frame reception from the MII, it only accepts from the start of the frame.

### (3) Flow Control

When the amount of receive data or the number of receive frames in the receive FIFO leads to one of the following conditions, the E-DMAC notifies the MAC to control MAC writing to the receive FIFO.

- When the space used in the receive FIFO exceeds the data amount specified by ETNAnFCFTRm
- When the number of receive frames in the receive FIFO exceeds the value specified by ETNAnFCFTRm

The threshold of the receive data amount can be set in a range from 256 to 65536 bytes in 256-byte units.

The threshold of receive frames can be set in a range from 1 to 24 frames (by the frame) in frame units.

### (4) Receive Descriptor Empty

When the RACT bit of the read descriptor is 0 (invalid), the receive descriptor empty state is determined and DMA transfer is stopped. Then the following operation is performed.

- Writes the RRm bit in ETNAnEDRRRm to 0
- Sets the RDE bit in ETNAnEESRm to 1 and generates an interrupt to the CPU.

To resume the DMA transfer to the receive buffer, the interrupt source needs to be cleared by software, the receive descriptor needs to be re-set and the RRm bit in ETNAnEDRRRm should be set to 1.

Even if receive descriptor is empty, frame reception from the MII to the receive FIFO is continued if there is empty space left in the receive FIFO and receive frame information management area. Therefore, even if a receive descriptor empty state is determined, the DMA transfer can be performed without discarding the frames received from the MII if DMA transfer to the receive buffer can be resumed before an overflow occurs.

### 21.4.33 CAM Function

Frames input to the MAC are grouped into the following four types; unicast for the product, broadcast, multicast, and unicast to other destinations. The MAC addresses of unicast for the product and broadcast are fixed, and determined only by register settings. Consequently, only multicast and unicast to other destinations determine whether to receive or not and whether to transfer or not by using the CAM (unicast frames whose destination MAC addresses match the product are called unicast frames to the product, and those that do not are called unicast frames to other destinations).

Furthermore, the evaluation of receive of unicast to other destinations and multicast frames by using CAM are performed by referencing the registered MAC addresses of the CAM entry table in the TSU. By using this function, receive FIFO overflow can be prevented caused by accumulation of frame data not required for reception, and CPU processing for determining receive can be reduced.

Each entry in the CAM table can be assigned to port 0, port 1 or both of them by using POST registers (ETNAnTSU_POST1 to 4). When the corresponding bit is set to 1, the CAM evaluation results are used for determining receive. In other words, when the corresponding bit of the POST table is cleared to 0, receive evaluation will be the same as when CAM is not used shown in **Table 21.87 Receive Frame Processing**

The on-chip CAM has entry tables which can register the MAC address of 32 entries, the details of which can be set by ETNAnTSU_ADRH0 to ENATnTSU_ADRH31 and ETNAnTSU_ADRL0 to ETNAnTSU_ADRL31. The setting to enable/disable referencing of the on-chip CAM entry table is performed by the CAM entry table enable setting register which sets whether to perform CAM evaluation or not, and the CAM entry table POST setting register for setting whether to use the CAM determination results for determining receive. When on-chip CAM entry table referencing during receive is enabled, the destination address in the frame and MAC address registered in the CAM entry table are compared, and it is determined whether to transfer the frames input to the MAC to E-DMAC (have E-DMAC receive the frames) or discard the frames. **Table 21.87 Receive Frame Processing** shows the processing method of frames (receive or discard) in reception from MAC0 to E-DMAC0 or that from MAC1 to E-DMAC1,

**Table 21.87 Receive Frame Processing**

| CAM Entry Table Referencing Results | Types of Frame | PRMm = 0 (Normal mode) | | PRMm = 1 (Promiscuous Mode) | |
|---|---|---|---|---|---|
| | | MCTm = 0 | MCTm = 1 | MCTm = 0 | MCTm = 1 |
| CAM hit | unicast for the product | Discarded | | Discarded | |
| | Broadcast | Discarded | | Discarded | |
| | Multicast | Discarded | Received | Discarded | Received |
| | unicast to other product | Received | | Discarded | |
| CAM mishit | unicast for the product | Received | | Received | |
| | Broadcast | Received | | Received | |
| | Multicast | Received | Discarded | Received | Discarded |
| | unicast to other product | Discarded | | Received | |
| CAM disable | unicast for the product | Same as CAM mishit | | | |
| | Broadcast | | | | |
| | Multicast | | | | |
| | unicast to other product | | | | |

[Legend]
PRMm (Bit 0 in ETNAnECMRm): Promiscuous Mode
(0: Normal mode, 1: Promiscuous Mode)
MCTm (Bit 13 in ETNAnECMRm):Multicast receive mode
(0: Receive when CAM mishit, 1: Receive when CAM hit)

## 21.4.34 Transmit/Receive Processing of Multi-Buffer Frame (Single-Frame/Multi-Descriptor)

### (1) Multi-Buffer Frame Transmit Processing in store and forward mode

During multi-buffer frame transmission, the processing shown in **Figure 21.16** is carried out by the E-DMAC.

In the figure the transmit descriptor is shown as active (TACT bit = 1), in case either buffer data has already been transmitted successfully or not. After all descriptors of multi-buffer frame are processed, frame transmission is started. If transmission complete interrupts are enabled in ENTAnEESIPRm, an interrupt is generated immediately after the final descriptor write-back at the end of frame transmission.

E-DMAC does descriptor write-back only to the first and last descriptors

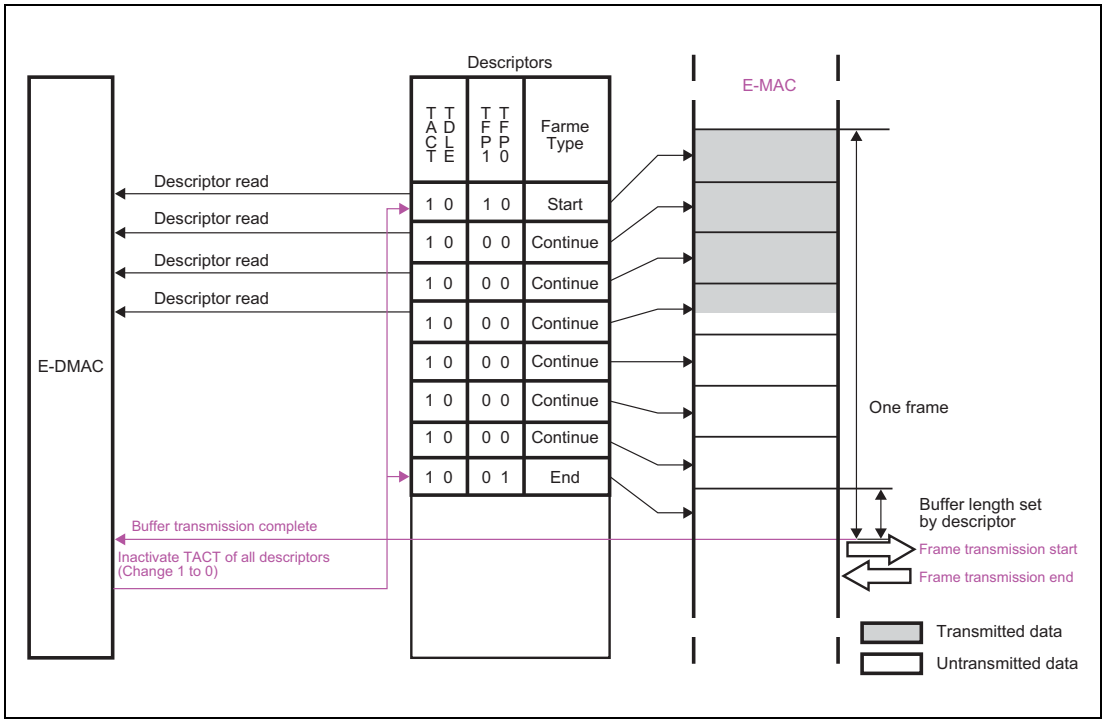



**Figure 21.16 E-DMAC Operation of multi-buffer frame transmission in store and forward mode**

### (2) Receive Processing in the Case of Multi-Buffer Frame

If an error occurs during reception in the case of a multi-buffer frame where a receive frame is divided for storage in multiple buffers, the E-DMAC performs the processing shown in **Figure 21.17, E-DMAC Operation after Receive Error**.

In the figure, the invalid receive descriptors (with the RACT bit cleared to 0) represent the successful reception of data to be stored in buffers, and the valid receive descriptors (with the RACT bit set to 1) represent unreceived buffers. If a frame receive error occurs with a descriptor shown in the figure, the status is written back to the corresponding descriptor.

If error interrupts are enabled in ETNAnEESIPRm, an interrupt is generated immediately after the write-back. If there is a new frame receive request, reception is continued from the buffer after that in which the error occurred.

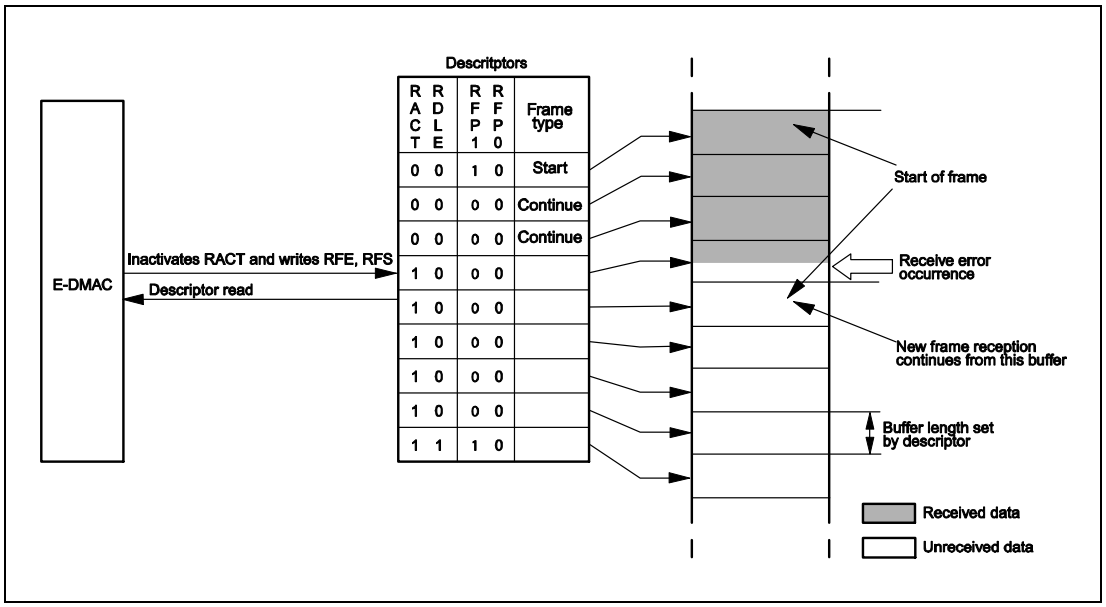



**Figure 21.17 E-DMAC Operation after Receive Error**

### 21.4.35 Padding Insertion in Receive Data

In the E-DMAC, 1 to 31 bytes of padding can be inserted in any byte position of receive data to improve software handling capability. By using this function, for instance, inserting 2-byte padding after the MAC header (14 bytes) of Ethernet frame enables data following the MAC header to set in 4-byte boundary.

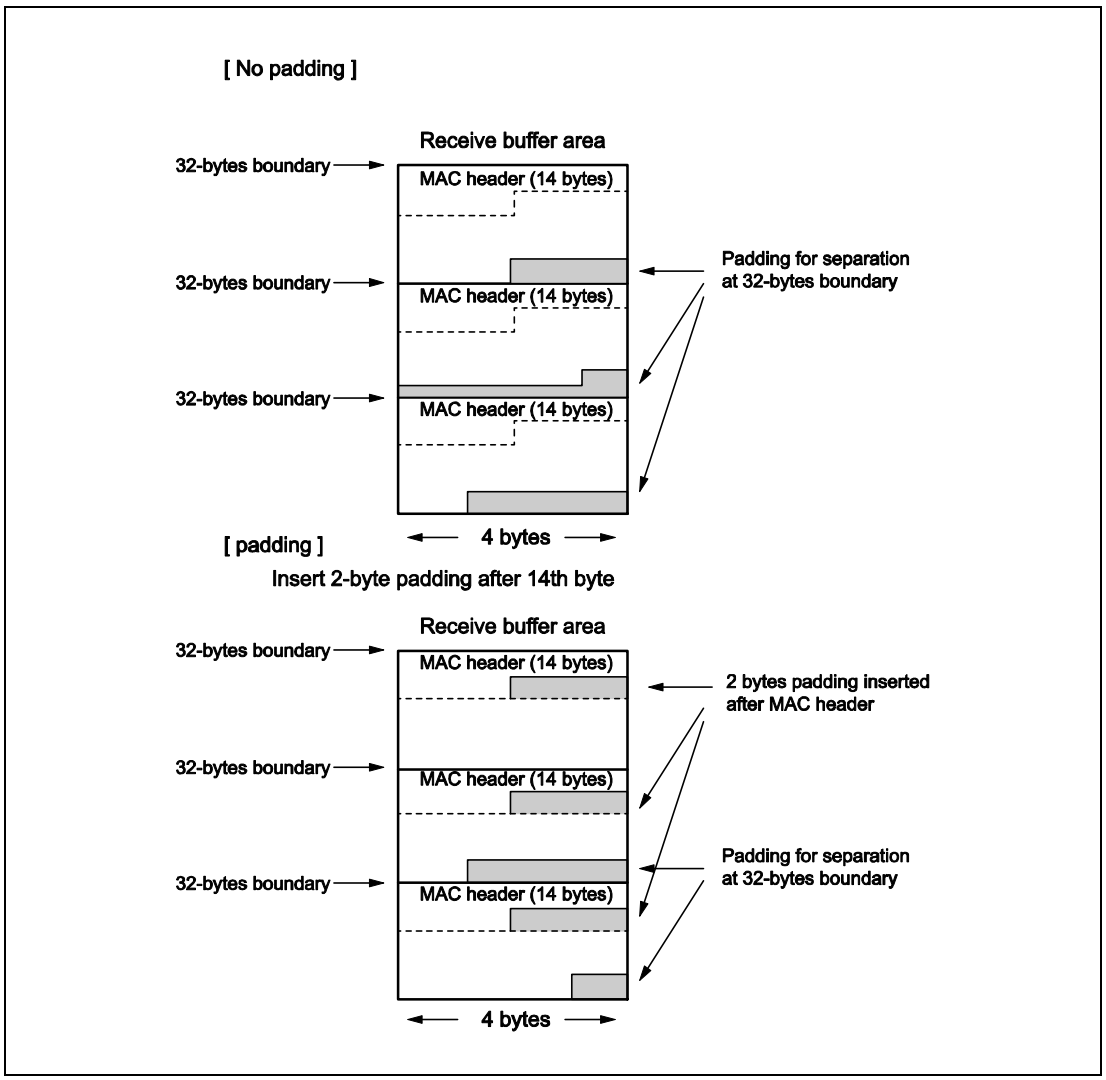



**Figure 21.18 Padding Insertion in Receive Data**

### 21.4.36 Interrupt Sources

The ETNA issues three types of interrupts: receive/transmit interrupts for port 0 (INTETNA0), receive/transmit interrupts for port 1 (INTETNA1) and detected status with transfer for TSU. **Table 21.88, List of ETNA Interrupts and detected statuses** shows these three interrupts, the interrupt sources, interrupt status registers/bits set at interrupt occurrence, and interrupt generation timing.

INTETNA0 or INTETNA1 interrupts are generated in correspondence with the port 0 or port 1 transmit/receive operation. When an interrupt source is generated, it is set in ETNAnEESR0 or ETNAnEESR1 and an interrupt is issued. For some interrupt sources, the ETNAnEESR0/ETNAnEESR1 setting and an interrupt are performed after a write-back operation to a descriptor is completed, not immediately after the interrupt source is detected. Interrupt sources other than the MAC status register source (ECIm bit) are cleared by writing a 1 to the corresponding source bit. The MAC status register source (ECIm bit) is cleared by writing a 1 to the corresponding source bit in ETNAnECSRm. Interrupt source bits retain the values until they are cleared. INTETNAn port 0 or port 1 interrupt source is allowed to issue interrupts by setting the corresponding bit in ETNAnEESIPR0 or ETNAnEESIPR1. Each MAC state register source (ECI bit) is allowed to issue an interrupt by setting the corresponding bit in ETNAnECSIPRm. In the value after reset, interrupts are disabled. Interrupt events generated inside E-MAC (e.g. link layer status) will be routed through E-DMAC, so that masking E-DMAC interrupt by HDIEm bit of ETNAnHDMIERm also masks any E-MAC interrupt event enabled in ETNAnECSIPRm.

**Table 21.88, List of ETNA Interrupts and detected statuses** shows these three interrupts, interrupt sources, interrupt status registers and bits set at interrupt occurrence and interrupt generation timing.

Table 21.88 List of ETNA Interrupts and detected statuses (1/3)

| Interrupt/ Status | Interrupt Source/Status | Register and Bit | Generation Timing |
|---|---|---|---|
| Transmit/ receive interrupt for port 0 (INTETNA0) | Write-back completed | ETNAnEESR0.TWB0 | After write-back |
| | Receive overflow frame write-back completed | ETNAnEESR0.ROC0 | After write-back |
| | Reception abort detection | ETNAnEESR0.RABT0 | After write-back |
| | Receive frame counter overflow | ETNAnEESR0.RFCOF0 | When the interrupt source is detected |
| | MAC status register source | ETNAnEESR0.ECI0 | When the interrupt source is detected |
| | Frame transmission completed | ETNAnEESR0.TC0 | After write-back |
| | Transmit descriptor empty | ETNAnEESR0.TDE0 | When the interrupt source is detected |
| | Frame reception | ETNAnEESR0.FR0 | After write-back |
| | Receive descriptor empty | ETNAnEESR0.RDE0 | When the interrupt source is detected |
| | Receive FIFO overflow | ETNAnEESR0.RFOF0 | When the interrupt source is detected |
| | Detect Loss of Carrier | ETNAnEESR0.DLC0 | When the interrupt source is detected |
| | Delayed Collision Detect | ETNAnEESR0.CD0 | When the interrupt source is detected |
| | Transmit Retry Over | ETNAnEESR0.TRO0 | When the interrupt source is detected |

Table 21.88 List of ETNA Interrupts and detected statuses (2/3)

| Interrupt/ Status | Interrupt Source/Status | Register and Bit | Generation Timing |
|---|---|---|---|
| Transmit/ receive interrupt for port 0 (INTETNA0) | Receive Multicast Address Frame | ETNAnEESR0.RMAF0 | After write-back |
| | Receive Residual-Bit Frame | ETNAnEESR0.RRF0 | After write-back |
| | Receive Too-Long Frame | ETNAnEESR0.RTLF0 | After write-back |
| | Receive Too-Short Frame | ETNAnEESR0.RTSF0 | After write-back |
| | PHY-LSI Receive Error | ETNAnEESR0.PRE0 | After write-back |
| | CRC Error on Received Frame | ETNAnEESR0.CERF0 | After write-back |
| Transmit/ receive interrupt for port 1 (INTETNA1) | Write-Back Completed | ETNAnEESR1.TWB1 | After write-back |
| | Receive Overflow Frame Write-Back Completed | ETNAnEESR1.ROC1 | After write-back |
| | Receive Abort Detect | ETNAnEESR1.RABT1 | After write-back |
| | Receive Frame Counter Overflow | ETNAnEESR1.RFCOF1 | When the interrupt source is detected |
| | MAC1 Status Register Source | ETNAnEESR1.ECI1 | When the interrupt source is detected |
| | Frame Transmission Completed | ETNAnEESR1.TC1 | After write-back |
| | Transmit Descriptor Empty | ETNAnEESR1.TDE1 | When the interrupt source is detected |
| | Frame Reception | ETNAnEESR1.FR1 | After write-back |
| | Receive Descriptor Empty | ETNAnEESR1.RDE1 | When the interrupt source is detected |
| | Receive FIFO Overflow | ETNAnEESR1.RFOF1 | When the interrupt source is detected |
| | Detect Loss of Carrier | ETNAnEESR1.DLC1 | When the interrupt source is detected |
| | Delayed Collision Detect | ETNAnEESR1.CD1 | When the interrupt source is detected |
| | Transmit Retry Over | ETNAnEESR1.TRO1 | When the interrupt source is detected |
| | Receive Multicast Address Frame | ETNAnEESR1.RMAF1 | After write-back |
| | Receive Residual-Bit Frame | ETNAnEESR1.RRF1 | After write-back |
| | Receive Too-Long Frame | ETNAnEESR1.RTLF1 | After write-back |
| | Receive Too-Short Frame | ETNAnEESR1.RTSF1 | After write-back |
| | PHY-LSI Receive Error | ETNAnEESR1.PRE1 | After write-back |
| | CRC Error on Received Frame | ETNAnEESR1.CERF1 | After write-back |

Table 21.88 List of ETNA Interrupts and detected statuses (3/3)

| Interrupt/ Status | Interrupt Source/Status | Register and Bit | Generation Timing |
|---|---|---|---|
| Detected status with transfer for TSU | MAC0 Residual-Bit Frame Receive | ETNAnTSU_FWSR.RINT50 | When the interrupt source is detected |
| | MAC0 Too-Long Frame Receive | ETNAnTSU_FWSR.RINT40 | When the interrupt source is detected |
| | MAC0 Too-Short Frame Receive | ETNAnTSU_FWSR.RINT30 | When the interrupt source is detected |
| | MAC0 Frame Receive Error | ETNAnTSU_FWSR.RINT20 | When the interrupt source is detected |
| | MAC0 CRC Error Frame Receive | ETNAnTSU_FWSR.RINT10 | When the interrupt source is detected |
| | MAC1 Residual-Bit Frame Receive | ETNAnTSU_FWSR.RINT51 | When the interrupt source is detected |
| | MAC1 Too-Long Frame Receive | ETNAnTSU_FWSR.RINT41 | When the interrupt source is detected |
| | MAC1 Too-Short Frame Receive | ETNAnTSU_FWSR.RINT31 | When the interrupt source is detected |
| | MAC1 Frame Receive Error | ETNAnTSU_FWSR.RINT21 | When the interrupt source is detected |
| | MAC1 CRC Error Frame Receive | ETNAnTSU_FWSR.RINT11 | When the interrupt source is detected |

## 21.4.37 Activation Procedure

This Ethernet module should be activated by the following procedure:

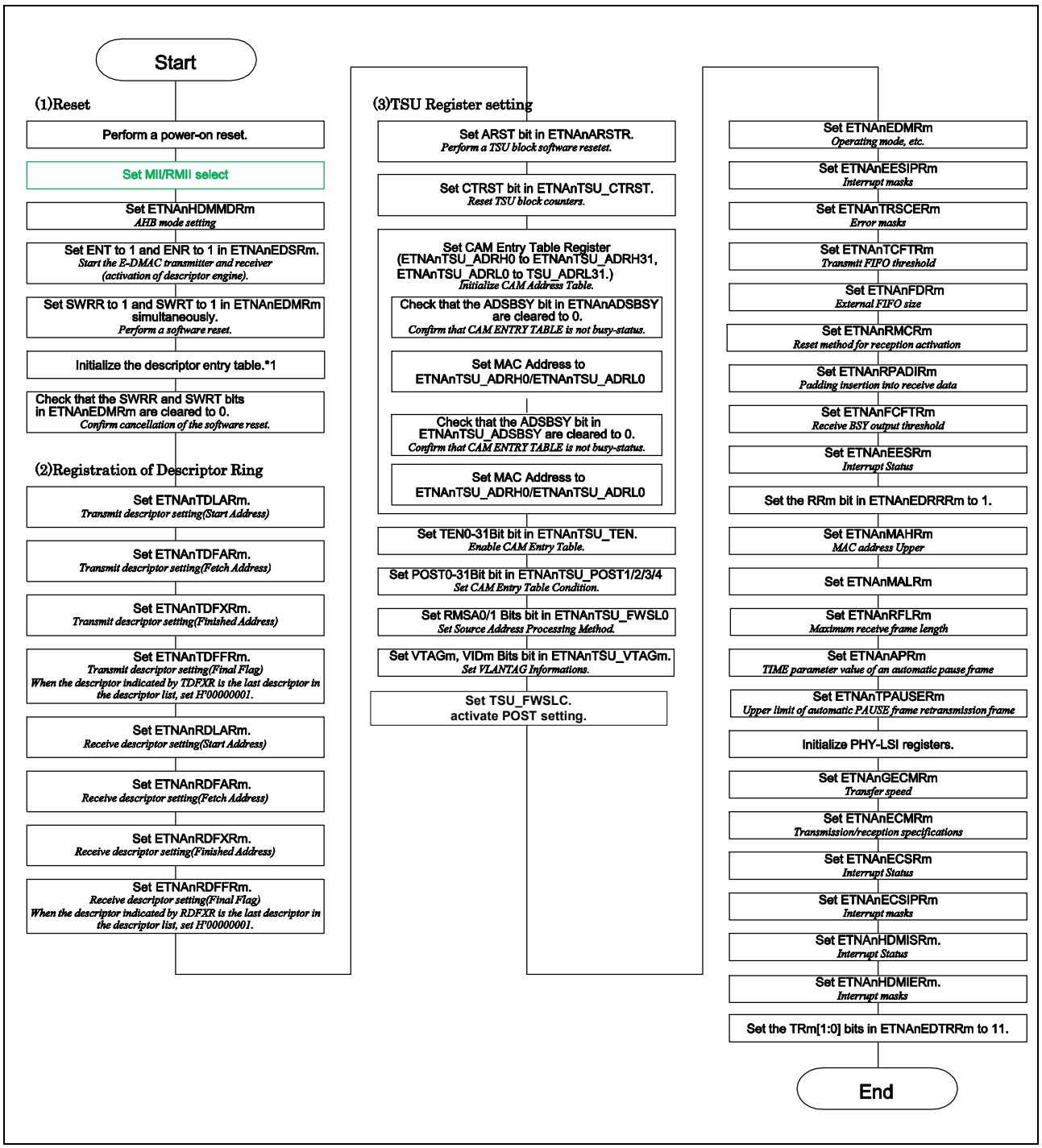



Figure 21.19 Transmit/Receive Start Procedure

**Note 1.** If the termination procedure which written in the next page is not conformed, the descriptor ring may be stopped halfway by unexpected write back which is caused by the transmission or the reception before the termination procedure.

## 21.4.38 Termination Procedure

The figure below shows the transmit/receive stop procedure.

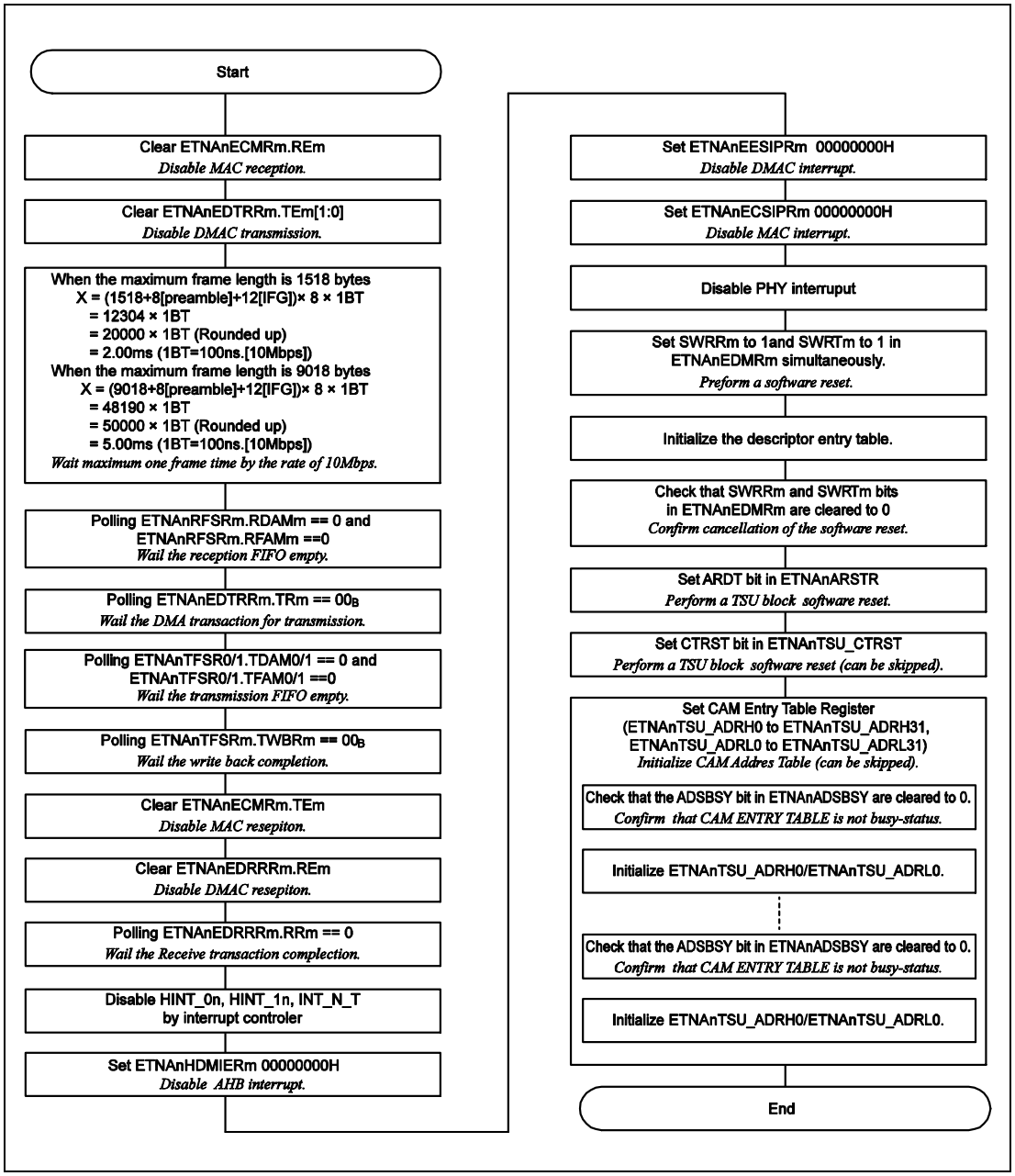



**Figure 21.20 Transmit Stop Procedure**

Even when a software reset for DMA transfer controller (E-DMAC) is issued, stop the transmission/ reception according to the procedure described above.

### 21.4.39 Flow Control

This Ethernet module supports flow control functions conforming to IEEE802.3x for full-duplex operation. The flow control can be applied to both receive and transmit operations. When transmitting PAUSE frames, flow control can be performed by the following two procedures.

### 21.4.40 Automatic PAUSE Frame Transmission

For receive frames, PAUSE frames are automatically transmitted when the number of data written to the receive FIFO reaches the value set in ETNAnFCFTRm. The TIME parameter included in the PAUSE frame is set by ETNAnAPRm. The automatic PAUSE frame transmission is repeated until the number of data in the receive FIFO becomes less than the value set in ETNAnFCFTRm as the receive data is read from the FIFO. Using ETNAnTPAUSERm, the upper limit of retransmission counts of the PAUSE frames can also be set in the range from 1 to 65535. In this case, PAUSE frame transmission is repeated until the number of receive FIFO data becomes less than the ETNAnFCFTRm value, or the number of transmits reaches the value set by ETNAnTPAUSERm. The transmission counter is cleared to 0 when the next PAUSE frame is transmitted after the number of data in the receive FIFO becomes less than the ETNAnFCFTRm value.

The automatic PAUSE frame transmission is enabled when the TXFm bit in ETNAnECMRm is 1.

### 21.4.41 Manual PAUSE Frame Transmission

PAUSE frames are transmitted by directives from the software. When writing the Timer value to ETNAnMPRm, manual PAUSE frame transmission is started. With this method, PAUSE frame transmission is carried out only once.

### 21.4.42 PAUSE Frame Reception

The next frame is not transmitted until the time indicated by the Timer value elapses after receiving a PAUSE frame. However, the transmission of the current frame is continued. A received PAUSE frame is valid only when the RXFm bit in ETNAnECMRm is set to 1. The number of times of PAUSE frame receptions is counted.

### 21.4.43 0-Time PAUSE Frame Control

Flow control is performed using a PAUSE frame with the TIME parameter value set to 0. The PAUSE frame with the TIME parameter set to 0 can be enabled or disabled by the ZPFm bit in ETNAnECMRm.

- When PAUSE frame control with the TIME parameter value set to 0 is enabled
  A PAUSE frame with the TIME parameter value set to 0 is transmitted when the number of data in the receive FIFO is less than the ETNAnFCFTRm value before the time indicated by the TIME parameter value has not elapsed. When a PAUSE frame with the time indicated by the TIME parameter value set to 0 is received, the transmit standby state is canceled.
- When PAUSE frame control with the TIME parameter value set to 0 is disabled
  A PAUSE frame with the TIME parameter value set to 0 is not transmitted. When a PAUSE frame with the TIME parameter value set to 0 is received, the PAUSE frame is discarded.

### 21.4.44 Back-Pressure Flow Control

Back-pressure mode is supported in half-duplex mode. In this mode, when the receive FIFO is full and can not be stored new frames, the following reception frame is forced to retry by the collision which is caused by dummy transmission of the Back Pressure.

### 21.4.45 Magic Packet Detection

This Ethernet module supports the Magic Packet format like shown below:

| DESTINATION (6byte) | SOURCE (6byte) | MISC (2byte~) | FF FF FF FF FF FF | MAC_ADDR×16 (ETNAnMAHRm/LRm) | MISC (0byte~) | CRC (4byte) |
|---|---|---|---|---|---|---|

This Ethernet module has a Magic Packet detection function. This function provides a Wake-On-LAN (WOL) facility that starts each peripheral device connected to a LAN from the host device or other source. This enables to construct a system in which a peripheral device receives a Magic Packet sent from the host device or other source, and starts itself. When the Magic Packet is detected, data is stored in the FIFO by the broadcast packet that has received data previously and the MAC is notified of the receiving status. To return to normal operation , the MAC, TSU and E-DMAC must be initialized by using ARSTm bit in ETNAnARSTR.

With a Magic Packet reception is performed regardless of the destination address in the frame header. Only in case the 16 times duplicated destination address contained in the Magic Packet Sequence matches the MAC address specified in the device, the CPU will be informed by the Magic Packet detection interrupt and the Wake-On-LAN (ETHmWOL) signal notifies that a Magic Packet has been detected. Further information on Magic Packets can be found in the technical documentation published by AMD Corporation.

The procedure for using the WOL function with this LSI is as follows.

1. Disable interrupt source output by means of the various interrupt enable/mask registers.
2. Set the MPDEm bit in ETNAnECMRm.
3. Set the MPDIP bit in ETNAnECSIPRm to the enable setting.
4. If necessary, set the chip operating mode to sleep mode or set peripheral modules to module standby mode.

When a Magic Packet is detected, an interrupt is sent to the system.

# 21.5 Connection to PHY-LSI

## 21.5.1 Accessing MII Registers

MII registers in the PHY-LSI are accessed via ETNAnPIRm. ETNAnPIRm is used as a serial interface conforming to the MII frame format specified in IEEE802.3u.

## 21.5.2 MII Management Frame Format

**Figure 21.21** shows the format of an MII management frame. To access an MII register, a management frame is implemented by the program in accordance with the procedures shown in MII Register Access Procedure.

| Access Type | MII Management Frame | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Item | PRE | ST | OP | PHYAD | REGAD | TA | DATA | IDLE |
| Bits | 32 | 2 | 2 | 5 | 5 | 2 | 16 | – |
| Read | 1..1 | 01 | 10 | 00001 | RRRRR | Z0 | D...D | – |
| Write | 1..1 | 01 | 01 | 00001 | RRRRR | 10 | D...D | X |

[Legend]
PRE: 32 consecutive 1
ST: Write of 01 indicating start of frame
OP: Write of code indicating access type
PHYAD: Write of 00001 when the PHY-LSI address is 1 (sequential write starting with the MSB)
The PHYAD bits vary with the PHY-LSI address.
REGAD: Write of 00001 when the register address is 1 (Sequential write starting with the MSB)
The REGAD bits vary with the PHY-LSI register address.
TA: Time for switching data transmission source on the MII interface
(a) Write: 10 written
(b) Read: Bus release [notation: Z0] performed
DATA: 16-bit data. Sequential write or read starting with the MSB
(a) Write: 16-bit data write
(b) Read: 16-bit data read
IDLE: Wait time until next MII management format input
(a) Write: Independent bus release [notation: X] performed
(b) Read: Bus already released at TA (control unnecessary)

**Figure 21.21 MII Management Frame Format**

### 21.5.3 MII Register Access Procedure

The program accesses MII registers via ETNAnPIRm. Access is implemented by a combination of 1-bit-unit data write, 1-bit-unit data read, bus release, and independent bus release. **Figure 21.22** shows the MII register access timing. The timing will differ depending on the PHY-LSI type.

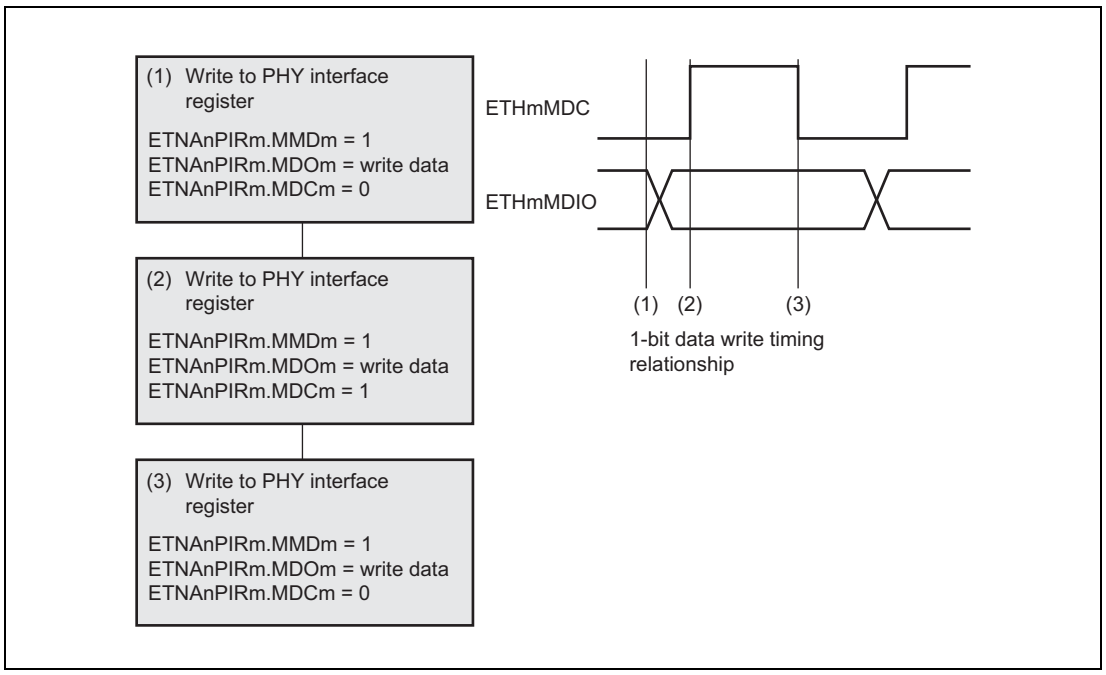



**Figure 21.22 1-Bit Data Write Flowchart**

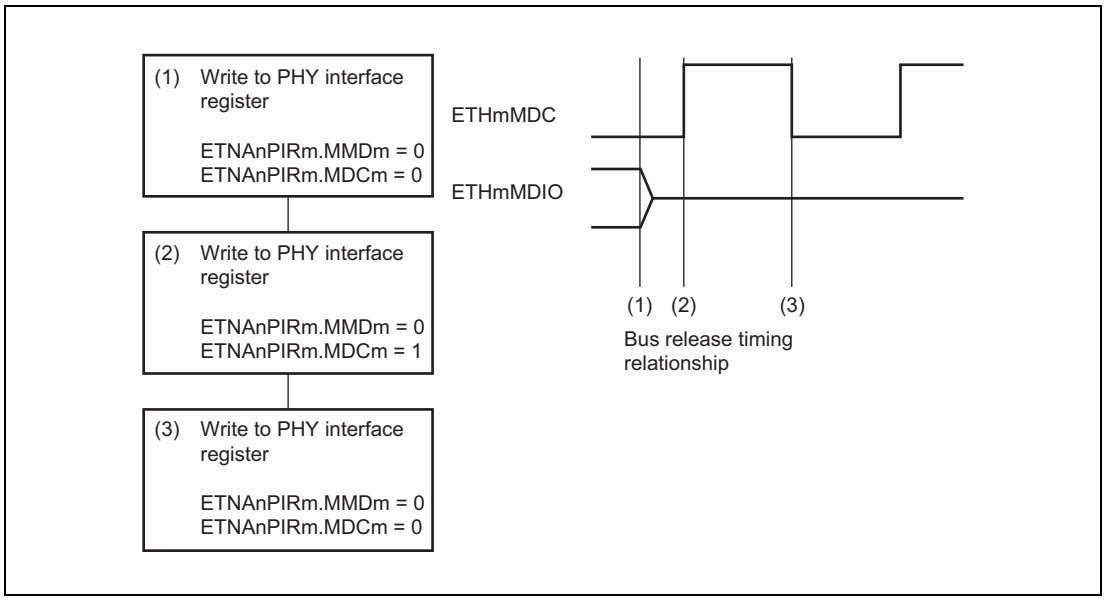



**Figure 21.23 Bus Release Flowchart (TA in Read in Figure 21.21)**

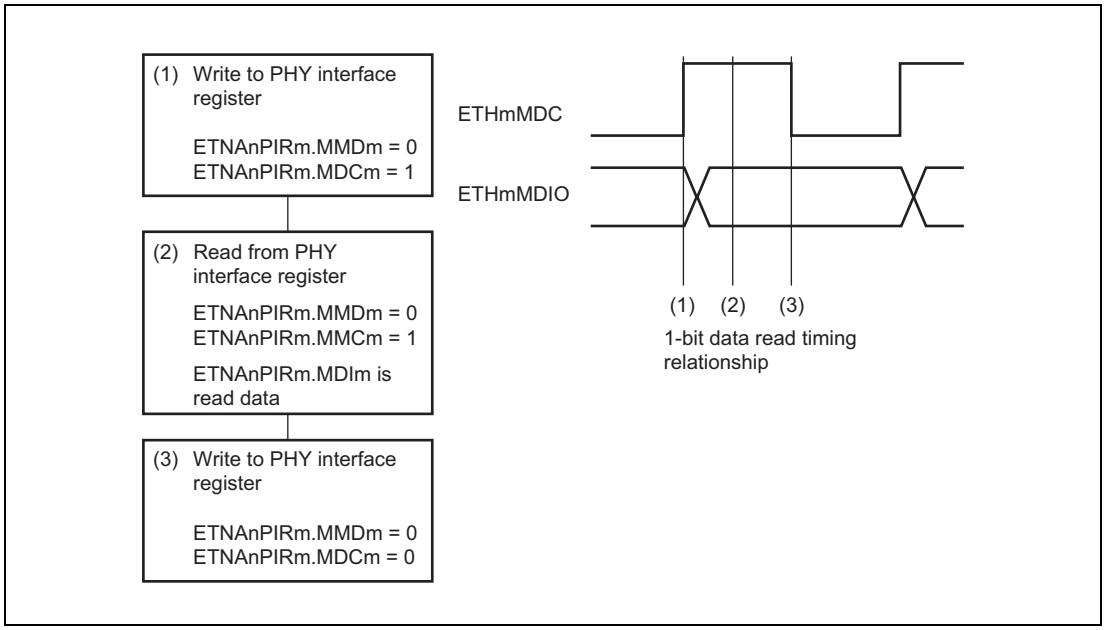



**Figure 21.24 1-Bit Data Read Flowchart**

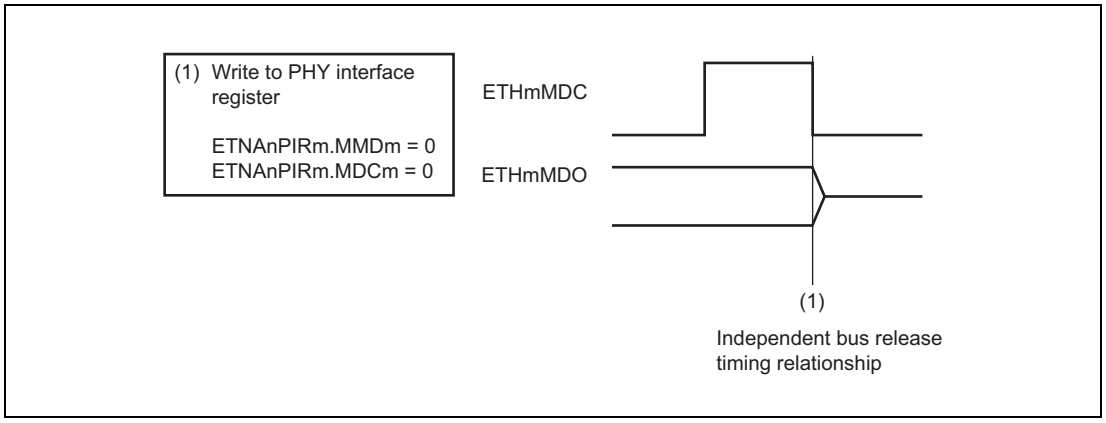



**Figure 21.25 Independent Bus Release Flowchart (IDLE in Write in Figure 21.21)**

## 21.6 Usage Notes

### 21.6.1 Checksum Calculation of Ethernet Frames

This LSI is capable of calculating the checksum data of the received frames. Only the data fields of the Ethernet frames are subject to calculation. Specifically, a data field follows the length/type field and is followed by the CRC field. Calculation involves 16-bit addition only; it does not involve bit reversal.

**NOTE**

Also for the frames with VLANTag inserted, the 15th byte from the top and the following bytes before the CRC field are subject to calculation.

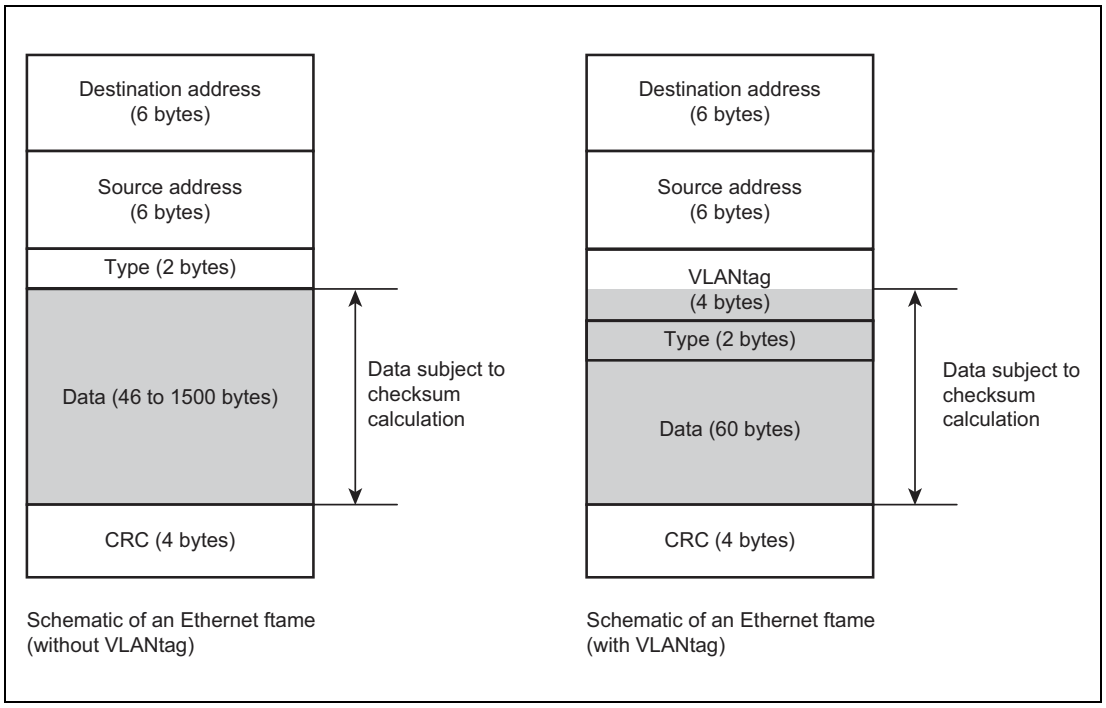



**Figure 21.26 Data Subject to Checksum Calculation**

## 21.7 Configuration of MII/RMII Mode

The bridge function of MII-to-RMII and RMII-to-MII is realized by converting the receive signal from RMII to MII and the transmit signal from MII to RMII by the MII2RMII block. The operation is performed in multiple different communication modes depending on the setting of the mode signal. **Table 21.89** lists the mode setting in each communication mode.

**Table 21.89 Operation State in Each Communication Mode**

| Communication mode | I/F specifications | MII | RMII | RMII | RMII | RMII |
|---|---|---|---|---|---|---|
| | Communication rate | 10 M/100 M | 10 Mbps | 10 Mbps | 100 Mbps | 100 Mbps |
| | Communication system | Full/half duplex | Half duplex | Full duiplex | Half duplex | Full duiplex |
| Register setting | ETNAnEDSRm.GPOm bit | 0 | 1 | | | |
| | ETNAnEMRm.SPEEDm1 bit | — | 0 | | 1 | |
| | ETNAnECMRm.DMm bit | — | 0 | 1 | 0 | 1 |

# Section 22 Single Edge Nibble Transmission (SENT)

This section describes the Renesas single edge nibble transmission (RSENT) module.

The first subsection describes all the characteristics specific to the RH850/P1x-C such as channels, register base addresses, and input/output signal names.
The second and subsequent subsections describe characteristics common to of SENT.

## 22.1 RH850/P1x-C SENT Overview

### 22.1.1 Number of Channels

The RH850/P1x-C includes up to ten SENT channels.

**Table 22.1 SENT Channels**

| | SENT |
|---|---|
| Number of Channels | n |
| Name | RSENTn |

#### Index n

This section identifies each RSENT channel by “n” (n = 0 to 9). For example, the RSENT timestamp register is described as RSENTnTSPC.

**Table 22.2 The number of channels in each product**

| Channel index | P1M-C (QFP, BGA-292) | P1M-C (BGA-156) | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|
| Channel number(n) | 6 (0 to 5) | 4 (0 to 3) | 8 (0 to 7) | 4 (0 to 3) | 8 (0 to 7) | 10 (0 to 9) |

### 22.1.2 Register Addresses

RSENT register addresses are represented by an offset from the base address <RSENTn_base>.

The following table shows the base address <RSENTn_base> of each RSENT module.

**Table 22.3 Register Base Address <RSENTn_base>**

| RSENTn Channel | <RSENTn_base> Address | P1M-C (QFP, BGA-292) | P1M-C (BGA-156) | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|---|
| RSENT0 | FFCD C000$_H$ | √ | √ | √ | √ | √ | √ |
| RSENT1 | FFCD C100$_H$ | √ | √ | √ | √ | √ | √ |
| RSENT2 | FFCD C200$_H$ | √ | √ | √ | √ | √ | √ |
| RSENT3 | FFCD C300$_H$ | √ | √ | √ | √ | √ | √ |
| RSENT4 | FFCD C400$_H$ | √ | — | √ | — | √ | √ |
| RSENT5 | FFCD C500$_H$ | √ | — | √ | — | √ | √ |
| RSENT6 | FFCD C600$_H$ | — | — | √ | — | √ | √ |
| RSENT7 | FFCD C700$_H$ | — | — | √ | — | √ | √ |
| RSENT8 | FFCD C800$_H$ | — | — | — | — | — | √ |
| RSENT9 | FFCD C900$_H$ | — | — | — | — | — | √ |

### 22.1.3 Clock Supply

The following clock input is supplied for the RSENT module.

**Table 22.4 Clock Supply**

| Unit Name | Clock for the Unit | Supply Clock Name |
|---|---|---|
| RSENTn | P-Bus interface clock (PCLK) | CLK_LSB |
| | RSENT communication clock(CLKC) | CLKP_H1 |

**Note:** CLK_LSB frequency can be in the range of 16MHz to 40MHz.
CLKP_H1 frequency can be either 16MHz or in the range of 32MHz to 80MHz.

For detail of clock supply, see **Section 12, Clock Controller**.

### 22.1.4 Interrupts and DMA requests

The RSENT module can generate the following interrupt requests.

**Table 22.5 Interrupt Requests (1/2)**

| Unit Interrupt Name | Outline | Interrupt Number | DMA Trigger Number | DTS Trigger Number |
|---|---|---|---|---|
| **RSENT0** | | | | |
| INTSENT0SI | RSENT status interrupt | 208 | — | — |
| INTSENT0RI | RSENT receive interrupt | 209 | 98 | 108 |
| **RSENT1** | | | | |
| INTSENT1SI | RSENT status interrupt | 210 | — | — |
| INTSENT1RI | RSENT receive interrupt | 211 | 99 | 109 |
| **RSENT2** | | | | |
| INTSENT2SI | RSENT status interrupt | 212 | — | — |
| INTSENT2RI | RSENT receive interrupt | 213 | 100 | 110 |
| **RSENT3** | | | | |
| INTSENT3SI | RSENT status interrupt | 214 | — | — |
| INTSENT3RI | RSENT receive interrupt | 215 | 101 | 111 |
| **RSENT4** | | | | |
| INTSENT4SI | RSENT status interrupt | 216 | — | — |
| INTSENT4RI | RSENT receive interrupt | 217 | 102 | 112 |
| **RSENT5** | | | | |
| INTSENT5SI | RSENT status interrupt | 218 | — | — |
| INTSENT5RI | RSENT receive interrupt | 219 | 103 | 113 |
| **RSENT6** | | | | |
| INTSENT6SI | RSENT status interrupt | 220 | — | — |
| INTSENT6RI | RSENT receive interrupt | 221 | 104 | 114 |
| **RSENT7** | | | | |
| INTSENT7SI | RSENT status interrupt | 222 | — | — |
| INTSENT7RI | RSENT receive interrupt | 223 | 105 | 115 |
| **RSENT8** | | | | |
| INTSENT8SI | RSENT status interrupt | 224 | — | — |
| INTSENT8RI | RSENT receive interrupt | 225 | 106 | 116 |

Table 22.5 Interrupt Requests (2/2)

| Unit Interrupt Name | Outline | Interrupt Number | DMA Trigger Number | DTS Trigger Number |
|---|---|---|---|---|
| **RSENT9** | | | | |
| INTSENT9SI | RSENT status interrupt | 226 | — | — |
| INTSENT9RI | RSENT receive interrupt | 227 | 107 | 117 |

## 22.1.5 RSENT Hardware Reset

The RSENT module and its registers are initialized by the following reset signal.

Table 22.6 Register Reset Condition

| Unit Name | Register Name | Reset Condition | | | | |
|---|---|---|---|---|---|---|
| | | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| RSENTn | All registers | √ | √ | √ | √ | √ |

**Note:** See **Section 8, Reset Controller**.
And RSENTn have RESET mode, see **Section 22.4.1.1, RESET Mode**.

### 22.1.6 External Input/Output Signals

The following table shows the RSENT input/output signals.

**Table 22.7 RSENTn Input/Output Signals**

| RSENTn Signal | Function | Connected to |
|---|---|---|
| **RSENT0** | | |
| sent_rx | RSENT data input | Port SENT0RX |
| sent_spc | RSENT SPC extension output | Port SENT0SPCO |
| **RSENT1** | | |
| sent_rx | RSENT data input | Port SENT1RX |
| sent_spc | RSENT SPC extension output | Port SENT1SPCO |
| **RSENT2** | | |
| sent_rx | RSENT data input | Port SENT2RX |
| sent_spc | RSENT SPC extension output | Port SENT2SPCO |
| **RSENT3** | | |
| sent_rx | RSENT data input | Port SENT3RX |
| sent_spc | RSENT SPC extension output | Port SENT3SPCO |
| **RSENT4** | | |
| sent_rx | RSENT data input | Port SENT4RX |
| sent_spc | RSENT SPC extension output | Port SENT4SPCO |
| **RSENT5** | | |
| sent_rx | RSENT data input | Port SENT5RX |
| sent_spc | RSENT SPC extension output | Port SENT5SPCO |
| **RSENT6** | | |
| sent_rx | RSENT data input | Port SENT6RX |
| sent_spc | RSENT SPC extension output | Port SENT6SPCO |
| **RSENT7** | | |
| sent_rx | RSENT data input | Port SENT7RX |
| sent_spc | RSENT SPC extension output | Port SENT7SPCO |
| **RSENT8** | | |
| sent_rx | RSENT data input | Port SENT8RX |
| sent_spc | RSENT SPC extension output | Port SENT8SPCO |
| **RSENT9** | | |
| sent_rx | RSENT data input | Port SENT9RX |
| sent_spc | RSENT SPC extension output | Port SENT9SPCO |

## 22.2 Overview

### Overview of Functions

The RSENT interface supports the following standard specification (SAE J2716 version JAN2010) functions:

- Triple speed expansion Tick Time: Clock cycle (1 µs to 90 µs)
- Variable data transmission rate
  - 24.7 kbps to 64.9 kbps: 3 µs clock rate 6 nibble data
  - 74.1 kbps to 194.8 kbps: 1 µs clock rate 6 nibble data
- Unidirectional communication: Between the sensor and MCU
- Bidirectional communication: Between the sensor and MCU (supported in SPC mode)
- Single edge data transmission: Coded by the temporal distance of two serially-detected falling edges on a data line
- Transmission frame with up to 6 data nibbles and additional status and communication nibble
- Data transmission protected with CRC is available.
  - CRC data can be read with the RSENTnSRXD.SCRC and RSENTnFRXD.FCRC bits.
- Calibration phase in each data frame (RSENTnCPL.CPLV bits)
- Multiple sensors can be connected to one RSENT channel by means of the SPC function.
- Interrupt functions: Receive interrupt and status interrupt.
- DMA function: Receive interrupt is used for DMA trigger
  (DMA line is same as receive interrupt line.)
- The timestamp function: Master or slave can be selected for the RSENT module. (RSENTnTSPC.TMS bit)

Each RSENT macro consists of one RSENT channel.

Each time stamp counter of the macros can run independently, or in order to synchronise the timestamp across multiple channels, one instance can be set as master and reset the configured consecutive time stamp counters. Depending on the total number of instances, more than one master-slave(s) pairs can be configured.

Each RSENT macro supports below requirements in addition to SAE J2716 specification of JAN2010:

R1: 32-Bit-Register for serial sensor data (24Bit + CRC6 + READ-Bit)

R2: Includes comparator for sensor data evaluation

R3: CRC check of received sensor data implemented but CRC code transparent

R4: 32-Bit counter for time stamp (resolution: 1µs)

R5: clock ticks down to 1µs

R6: SPC (Short PWM Code) extension

- Enables bidirectional communication channel
- Master can pull down the signal to initiate SENT message

- The RSENT circuit consists of the following functions:
  - Data receive part
  - Clock recovery
  - Register group

### 22.2.1 Block Diagram

The following figure shows a block diagram of the RSENT module.

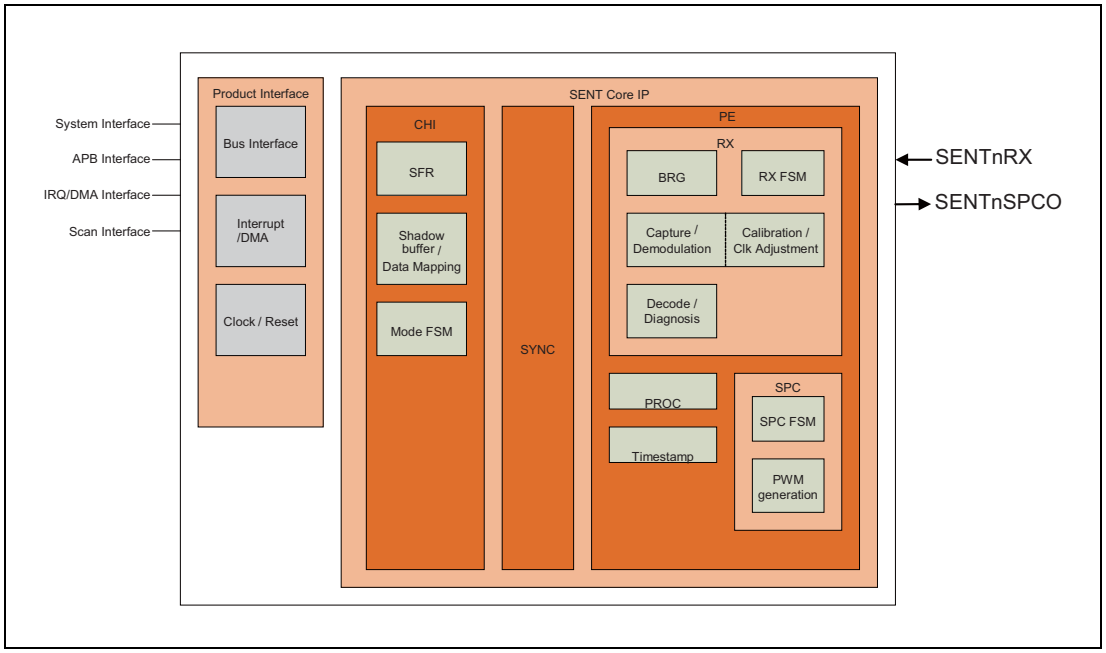



**Figure 22.1 Block Diagram of RSENT**

## 22.3 Registers

RSENTn is controlled and operated by the following registers.

**Table 22.8 Overview of RSENTn Registers**

| Register Name | Abbreviation | Address | Access Protection | |
|---|---|---|---|---|
| | | | PBG | Other |
| RSENT timestamp register | RSENTnTSPC | <RSENTn_base> + $0000_H$ | PBG4#1.PG4-SENT | — |
| RSENT timestamp counter | RSENTnTSC | <RSENTn_base> + $0004_H$ | PBG4#1.PG4-SENT | — |
| RSENT communication configuration register | RSENTnCC | <RSENTn_base> + $0010_H$ | PBG4#1.PG4-SENT | — |
| RSENT baud rate prescaler register | RSENTnBRP | <RSENTn_base> + $0014_H$ | PBG4#1.PG4-SENT | — |
| RSENT interrupt/DMA enable register | RSENTnIDE | <RSENTn_base> + $0018_H$ | PBG4#1.PG4-SENT | — |
| RSENT mode control register | RSENTnMDC | <RSENTn_base> + $001C_H$ | PBG4#1.PG4-SENT | — |
| RSENT SPC transmission register | RSENTnSPCT | <RSENTn_base> + $0020_H$ | PBG4#1.PG4-SENT | — |
| RSENT mode status register | RSENTnMST | <RSENTn_base> + $0024_H$ | PBG4#1.PG4-SENT | — |
| RSENT communication status register | RSENTnCS | <RSENTn_base> + $0028_H$ | PBG4#1.PG4-SENT | — |
| RSENT communication status clear register | RSENTnCSC | <RSENTn_base> + $002C_H$ | PBG4#1.PG4-SENT | — |
| RSENT slow channel receive timestamp register | RSENTnSRTS | <RSENTn_base> + $0030_H$ | PBG4#1.PG4-SENT | — |
| RSENT slow channel receive data register | RSENTnSRXD | <RSENTn_base> + $0034_H$ | PBG4#1.PG4-SENT | — |
| RSENT calibration pulse length register | RSENTnCPL | <RSENTn_base> + $0038_H$ | PBG4#1.PG4-SENT | — |
| RSENT message length register | RSENTnML | <RSENTn_base> + $003C_H$ | PBG4#1.PG4-SENT | — |
| RSENT fast channel receive timestamp register | RSENTnFRTS | <RSENTn_base> + $0040_H$ | PBG4#1.PG4-SENT | — |
| RSENT fast channel receive data register | RSENTnFRXD | <RSENTn_base> + $0044_H$ | PBG4#1.PG4-SENT | — |
| RSENT0 timestamp mode selection register | RSENT0TSSEL | $FFCDCF00_H$ | PBG4#1.PG4-SENT | — |
| RSENT1 timestamp mode selection register | RSENT1TSSEL | $FFCDCF04_H$ | PBG4#1.PG4-SENT | — |
| RSENT2 timestamp mode selection register | RSENT2TSSEL | $FFCDCF08_H$ | PBG4#1.PG4-SENT | — |
| RSENT3 timestamp mode selection register | RSENT3TSSEL | $FFCDCF0C_H$ | PBG4#1.PG4-SENT | — |
| RSENT4 timestamp mode selection register | RSENT4TSSEL | $FFCDCF10_H$ | PBG4#1.PG4-SENT | — |
| RSENT5 timestamp mode selection register | RSENT5TSSEL | $FFCDCF14_H$ | PBG4#1.PG4-SENT | — |
| RSENT6 timestamp mode selection register | RSENT6TSSEL | $FFCDCF18_H$ | PBG4#1.PG4-SENT | — |
| RSENT7 timestamp mode selection register | RSENT7TSSEL | $FFCDCF1C_H$ | PBG4#1.PG4-SENT | — |
| RSENT8 timestamp mode selection register | RSENT8TSSEL | $FFCDCF20_H$ | PBG4#1.PG4-SENT | — |
| RSENT9 timestamp mode selection register | RSENT9TSSEL | $FFCDCF24_H$ | PBG4#1.PG4-SENT | — |

**<RSENTn_base>**

The base address <RSENTn_base> of RSENTn is defined by the keywords in **Table 22.3, Register Base Address <RSENTn_base>**.

### 22.3.1 RSENTnTSPC — RSENT Timestamp Register

**Access:** This register can be read/written in 32-bit units.

**Address:** <RSENTn_base> + $0000_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TMS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TTM[6:0] | | | | | | | — | TTPV[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 22.9 RSENTnTSPC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 16 | TMS | Timestamp Mode Selection<br>0: Master mode<br>1: Slave mode |
| 15 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 14 to 8 | TTM[6:0] | Timestamp Tick Multiplier<br>$0000000_B$: 1<br>$0000001_B$: 2<br>$0000010_B$: 3<br>:<br>$1111111_B$: 128 |
| 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 to 0 | TTPV[6:0] | Timestamp Tick Prescaler Value<br>$0000000_B$: 1<br>$0000001_B$: 2<br>$0000010_B$: 3<br>:<br>$1111111_B$: 128 |

#### RSENTnTSPC.TMS (Timestamp Mode Selection)

This bit defines the timestamp counter synchronization mode.

For information about the timestamp clock settings, see **Section 22.4.2.1, Timestamp**.

When this bit is set to 0, the timestamp counter operates in master mode.

When writing $0000\ 0000_H$ to RSENTnTSC, the timestamp counter is cleared. In addition all RSENT timestamp counters operating as slave of RSENT are also cleared. For master-slave interconnection, see **Section 22.4.2.1(2), Timestamp Counter Operation**.

When this bit is set to 1, the timestamp counter operates in slave mode.

The timestamp counter is only cleared when writing $0000\ 0000_H$ to the timestamp counter of RSENT module that operates in master mode.

The CPU can only write to this bit if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are 001).

The RSENT module operating in slave mode should have the same timestamp counter prescaler settings as the RSENT module that operates in master mode.

The CPU should not set this bit to 1 for RSENT module that operates in master mode.

#### RSENTnTSPC.TTM (Timestamp Tick Multiplier)

These bits define the multiplication value of the 1-µs time tick used for the timestamp counter.

For timestamp clock configuration, see **Section 22.4.2.1, Timestamp**.

The CPU can only write to these bits if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are 001).

#### RSENTnTSPC.TTPV (Timestamp Tick Prescaler Value)

These bits define the prescaler value to generate a 1-µs clock tick.

For timestamp clock configuration, see **Section 22.4.2.1, Timestamp**.

The CPU can only write to these bits if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are 001).

The CPU should configure this value in such a way that, based on the supplied communication clock, a 1-µs clock tick is generated.

## 22.3.2 RSENTnTSC — RSENT Timestamp Counter

**Access:** This register can be read/written in 32-bit units.

**Address:** <RSENTn_base> + $0004_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TS[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 22.10 RSENTnTSC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TS[31:0] | Timestamp counter value |

### RSENTnTSC.TS (Timestamp)

These bits indicate the current timestamp counter value.

The CPU can write to these bits values other than $0000\ 0000_H$ only if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are 001).Even if any value is written in these bits, these bits are initialized.

When the timestamp counter is configured to operate in slave mode (RSENTnTSPC.TMS = 1), writing to this register has no effect when the RSENT module is in either of the OPERATION IDLE or OPERATION ACTIVE modes (the RSENTnMST.OMS bits are either 011 or 101).

The timestamp counter is incremented on every timestamp counter tick (as configured in the RSENTnTSPC.TTPV and RSENTnTSPC.TTM bits) when the RSENT module is in either of the OPERATION IDLE or OPERATION ACTIVE modes (the RSENTnMST.OMS bits are either 011 or 101).

When the timestamp counter is configured to operate in master mode (RSENTnTSPC.TMS = 0), the CPU writes $0000\ 0000_H$ to these bits and RSENTnTSC.TS is set to $0000\ 0000_H$.

When the timestamp counter is configured to operate in slave mode (RSENTnTSPC.TMS = 1), the CPU writes $0000\ 0000_H$ to the RSENTnTSC.TS bits of RSENT module that operates in master mode and the RSENTnTSC.TS bits are set to $0000\ 0000_H$.

### 22.3.3 RSENTnCC — RSENT Communication Configuration Register

**Access:** This register can be read/written in 32-bit units.

**Address:** <RSENTn_base> + $0010_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | SOPC | FCM | SCCD | FCCD | DFC | SMF[1:0] | | PPTC | PPC | NDN[2:0] | | | SPCE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 22.11 RSENTnCC register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 13 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 12 | SOPC | SPC Output Polarity Control<br>0: SPC pulse active high<br>1: SPC pulse active low |
| 11 | FCM | Frame Check Method<br>0: Check against next calibration pulse<br>1: Check against previous calibration pulse |
| 10 | SCCD | Slow Channel CRC Check<br>0: Slow channel CRC check enabled<br>1: Slow channel CRC check disabled |
| 9 | FCCD | Fast Channel CRC Check<br>0: Fast channel CRC check enabled<br>1: Fast channel CRC check disabled |
| 8 | DCF | Data nibble CRC Format<br>0: SAE J2716 2010 format<br>1: pre SAE J2716 2010 format |
| 7, 6 | SMF | Serial Message Format<br>00: No serial message extraction<br>01: Short serial message format<br>10: Enhanced serial message format<br>11: Setting prohibited |
| 5 | PPTC | Pause Pulse Type Configuration<br>0: Pause pulse for variable message length<br>1: Pause pulse for fixed message length |
| 4 | PPC | Pause Pulse Configuration<br>0: Pause pulse absent<br>1: Pause pulse present |
| 3 to 1 | NDN[2:0] | Number of Data Nibbles<br>000: 1 data nibble<br>001: 2 data nibbles<br>010: 3 data nibbles<br>011: 4 data nibbles<br>100: 5 data nibbles<br>101: 6 data nibbles<br>Other than above: Setting prohibited |

Table 22.11 RSENTnCC register contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | SPEC | SPC Mode Enable<br>0: SPC mode disabled<br>1: SPC mode enabled |

### RSENTnCC.SOPC (SPC Output Polarity Control)

When this bit is set to 0, the SPC pulse is sent as an active high signal. The default output value is low level.

When this bit is set to 1, the SPC pulse is sent as an active low signal. The default output value is high level.

For SPC operation, see also **Section 22.4.4, SPC Function**.

The CPU can only write to this bit if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are 001).

**NOTE**

Any change to this bit from the default value becomes effective on the output value when entering the OPERATION_ACTIVE mode (MST.OMS is 3'b101). When entering RESET mode (MST.OMS is 3'b000), the output level is set to the default value (low level).

### RSENTnCC.FCM (Frame Check Method)

When this bit is set to 0, the current calibration pulse is compared to the next received calibration pulse.

The buffer update mechanism is operating according to the preferred option as described in SAE J2716 2010.

When this bit is set to 1, the current calibration pulse is compared to the previously received calibration pulse.

The buffer update mechanism is operating according to the second option as described in SAE J2716 2010 which should be only used if extra latency to process the second calibration pulse cannot be tolerated.

For buffer update timings, see also **Section 22.4.3.2(3), Fast Channel Message Reception**.

The CPU can only write to this bit if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are 001).

### RSENTnCC.SCCD (Slow Channel CRC Check Disable)

When this bit is set to 1, the CRC check for the slow channel is disabled. In this case, messages are stored in the slow channel message buffer with the received CRC.

When this bit is set to 1, the RSENTnCS.SCS bit is not set.

The CPU can only write to this bit if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are 001).

### RSENTnCC.FCCD (Fast Channel CRC Check Disable)

When this bit is set to 1, the CRC check for the fast channel is disabled. In this case, messages are stored in the fast channel message buffer with the received CRC.

When this bit is set to 1, the RSENTnCS.FCS bit is not set.

The CPU can only write to this bit if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are 001).

#### RSENTnCC.DCF (Data nibble CRC Format)

This bit selects between the SAE J2716 2010 data nibble CRC format and the legacy format.

When this bit is set to 0 the recommended CRC implementation according to SAE J2716 2010 section 5.4.2.2 is selected.

When this bit is set to 1 the legacy CRC implementation according to SAE J2716 2008 is selected.

The CPU can only write to this bit if the RSENT module is in the CONFIGURATION mode (RSENTnMST.OMS = $001_B$)

#### RSENTnCC.SMF (Serial Message Format)

These bits define the serial message format expected to be received for automatic extraction.

When these bits are set to 00, no serial message is extracted and the status and communication nibble are provided in the RSENTnSRXD register.

The CPU can only write to these bits if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are 001).

The CPU shall set these bits to 00 when RSENTnCC.SPCE is set to 1 and more than one sensor is connected to the RSENT module.

#### RSENTnCC.PPTC (Pause Pulse Type Configuration)

This bit defines the pause pulse type.

The CPU can only write to this bit if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are $001_B$).

The CPU should not set this bit to 1 when the RSENTnCC.PPC bit is set to 0.

#### RSENTnCC.PPC (Pause Pulse Configuration)

This bit defines the presence or absence of the pause pulse.

The CPU can only write to this bit if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are $001_B$).

#### RSENTnCC.NDN (Number of Data Nibbles)

These bits define the number of data nibbles included in an SENT message.

The CPU can only write to these bits if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are $001_B$).

#### RSENTnCC.SPCE (SPC Mode Enable)

This bit enables the SPC mode.

For details about SPC mode operation, see also **Section 22.4.4, SPC Function**.

The CPU can only write to this bit if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are $001_B$).

### 22.3.4 RSENTnBRP — RSENT Baud Rate Prescaler Register

**Access:** This register can be read/written in 32-bit units.

**Address:** <RSENTn_base> + $0014_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TTF[3:0] | | | | — | TTI[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | SCDV[6:0] | | | | | | | — | — | — | — | SCMV[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 22.12 BRP register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 27 to 24 | TTF[3:0] | Time Tick Fraction<br>0000: 0.0 µs<br>0001: 0.1 µs<br>0010: 0.2 µs<br>:<br>1000: 0.8 µs<br>1001: 0.9 µs<br>Other than above: Setting prohibited |
| 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22 to 16 | TTI[6:0] | Time Tick Integer<br>0000000: 1 µs<br>0000001: 2 µs<br>0000010: 3 µs<br>:<br>1011000: 89 µs<br>1011001: 90 µs<br>Other than above: Setting prohibited |
| 15 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 14 to 8 | SCDV[6:0] | Sample Clock Division Value<br>0000100: 5<br>Other than above: Setting prohibited |
| 7 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | SCMV[3:0] | Sample Clock Multiplication Value<br>0000: 1<br>Other than above: Setting prohibited |

#### RSENTnBRP.TTF (Time Tick Fraction)

These bits define the fractional part of the tick length in 0.1-µs granularity.

For tick length configuration, see **Section 22.4.2.2(2) RX and SPC Tick Settings**.

The CPU can only write to these bits if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are $001_B$).

#### RSENTnBRP.TTI (Time Tick Integer)

These bits define the integer part of the tick length.

For tick length configuration, see **Section 22.4.2.2(2) RX and SPC Tick Settings**.

The CPU can only write to these bits if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are $001_B$).

#### RSENTnBRP.SCDV (Sample Clock Division Value)

These bits define the division value for the sample clock generation logic.

For RSENTnBRP settings, see **Section 22.4.2.2(1) RX BRP Setting**.

The CPU can only write to these bits if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are $001_B$).

#### RSENTnBRP.SCMV (Sample Clock Multiplication Value)

These bits define the multiplication value for the sample clock generation logic.

For RSENTnBRP settings, see **Section 22.4.2.2(1) RX BRP Setting**.

The CPU can only write to these bits if the RSENT module is in the CONFIGURATION mode (the RSENTnMST.OMS bits are $001_B$).

### 22.3.5 RSENTnIDE — RSENT Interrupt/DMA Enable Register

**Access:** This register can be read/written in 32-bit units.

**Address:** <RSENTn_base> + $0018_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SEIE | SMIE | SCIE | NRIE | CVIE | CLIE | FNIE | FEIE | FMIE | FCIE | FRIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 22.13 RSENTnIDE register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 10 | SEIE | Slow Channel Encoding Error Interrupt Control<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 9 | SMIE | Slow Channel Message Lost Interrupt Control<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 8 | SCIE | Slow Channel CRC Error Interrupt Control<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 7 | NRIE | No Response Error Interrupt Control<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 6 | CVIE | Calibration Pulse Length Variation Error Interrupt Control<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 5 | CLIE | Calibration Pulse Length Error Interrupt Control<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 4 | FNIE | Fast Channel Nibble Count Error Interrupt Control<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 3 | FEIE | Fast Channel Nibble Encoding Error Interrupt Control<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 2 | FMIE | Fast Channel Message Lost Interrupt Control<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 1 | FCIE | Fast Channel CRC Error Interrupt Control<br>0: Interrupt disabled<br>1: Interrupt enabled |
| 0 | FRIE | Fast Channel Receive Interrupt Control<br>0: Interrupt disabled<br>1: Interrupt enabled |

#### RSENTnIDE.SEIE (Slow Channel Encoding Error Interrupt Control)

This bit enables the generation of the slow channel encoding error interrupt.

The CPU cannot write to this bit if the RSENT module is in the RESET mode (the RSENTnMST.OMS bits are $000_B$).

#### RSENTnIDE.SMIE (Slow Channel Message Lost Interrupt Control)

This bit enables the generation of the slow channel message lost interrupt.

The CPU cannot write to this bit if the RSENT module is in the RESET mode (the RSENTnMST.OMS bits are $000_B$).

#### RSENTnIDE.SCIE (Slow Channel CRC Error Interrupt Control)

This bit enables the generation of the slow channel CRC error interrupt.

The CPU cannot write to this bit if the RSENT module is in the RESET mode (the RSENTnMST.OMS bits are $000_B$).

#### RSENTnIDE.NRIE (No Response Error Interrupt Control)

This bit enables the generation of the no response error interrupt.

The CPU cannot write to this bit if the RSENT module is in the RESET mode (the RSENTnMST.OMS bits are $000_B$).

The CPU should not set this bit when the SPC mode is disabled (RSENTnCC.SPCE set to 0).

#### RSENTnIDE.CVIE (Calibration Pulse Length Variation Error Interrupt Control)

This bit enables the generation of the calibration pulse length variation error interrupt.

The CPU cannot write to this bit if the RSENT module is in the RESET mode (the RSENTnMST.OMS bits are $000_B$).

#### RSENTnIDE.CLIE (Calibration Pulse Length Error Interrupt Control)

This bit enables the generation of the calibration pulse length error interrupt.

The CPU cannot write to this bit if the RSENT module is in the RESET mode (the SENTnMST.OMS bits are $000_B$).

#### RSENTnIDE.FNIE (Fast Channel Nibble Count Error Interrupt Control)

This bit enables the generation of the fast channel nibble count error interrupt.

The CPU cannot write to this bit if the RSENT module is in the RESET mode (the RSENTnMST.OMS bits are $000_B$).

#### RSENTnIDE.FEIE (Fast Channel Nibble Encoding Error Interrupt Control)

This bit enables the generation of the fast channel nibble encoding error interrupt.

The CPU cannot write to this bit if the RSENT module is in the RESET mode (the SENTnMST.OMS bits are $000_B$).

#### RSENTnIDE.FMIE (Fast Channel Message Lost Interrupt Control)

This bit enables the generation of the fast channel message lost interrupt.

The CPU cannot write to this bit if the RSENT module is in the RESET mode (the SENTnMST.OMS bits are $000_B$).

#### RSENTnIDE.FCIE (Fast Channel CRC Error Interrupt Control)

This bit enables the generation of the fast channel CRC error interrupt.

The CPU cannot write to this bit if the RSENT module is in the RESET mode (the RSENTnMST.OMS bits are $000_B$).

#### RSENTnIDE.FRIE (Fast Channel Receive Interrupt Control)

This bit enables the generation of the fast channel receive interrupt.

The fast channel receive interrupt can be also used to notify a DMA request.

The CPU cannot write to this bit if the RSENT module is in the RESET mode (the RSENTnMST.OMS bits are $000_B$).

## 22.3.6 RSENTnMDC — RSENT Mode Control Register

**Access:** This register can be read/written in 32-bit units.

**Address:** <RSENTn_base> + $001C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | OMC[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 22.14 RSENTnMDC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 2 to 0 | OMC[2:0] | Operation Mode Control<br>000: RESET<br>001: CONFIGURATION<br>011: OPERATION IDLE<br>101: OPERATION ACTIVE<br>Other than above: Setting prohibited |

### RSENTnMDC.OMC (Operation Mode Control)

These bits are used to control the operation mode of the RSENT module.

- 000: RESET

  In RESET mode, the mode can only be changed to CONFIGURATION mode.

- 001: CONFIGURATION

  In CONFIGURATION mode, the mode can only be changed to RESET mode or OPERATION ACTIVE mode.

- 011: OPERATION IDLE

  In OPERATION IDLE mode, the mode can be changed to OPERATION ACTIVE mode, CONFIGURATION mode, or RESET mode.

- 101: OPERATION ACTIVE

  In OPERATION ACTIVE mode, the mode can be changed to OPERATION IDLE mode, CONFIGURATION mode, or RESET mode. However, it is recommended to process to the OPERATION IDLE mode first.

  For the recommended methods to change between operation modes, see **Section 22.4.3.1, Changing Operation Modes**.

- Other than above: Setting prohibited

The CPU should not write any other value than listed above into this register.
The CPU should follow the mode change flows as shown in section **Section 22.4.3.1, Changing Operation Modes**.

**NOTE**

When the CPU is requesting a not supported mode change, writing to this register has no effect.

## 22.3.7 RSENTnSPCT — RSENT SPC Transmission Register

**Access:** This register can be read/written in 32-bit units.

**Address:** <RSENTn_base> + $0020_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | TLL[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 22.15 RSENTnSPCT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 to 0 | TLL[6:0] | Length of the Trigger Low Phase in Ticks<br>0000000: 1 tick<br>0000001: 2 ticks<br>0000010: 3 ticks<br>:<br>1111110: 127 ticks<br>1111111: 128 ticks |

### RSENTnSPCT.TLL (Trigger Low Length)

These bits define the length of the SPC trigger pulse.

When the CPU writes to these bits, an SPC trigger pulse with the configured length is sent starting from the next SPC trigger tick. In case RSENTnCS.NRS is set by the RSENT module following a write to these bits, noSPC trigger pulse is sent..

For details about SPC communication, see **Section 22.4.4, SPC Function**.

The CPU can only write to these bits if the RSENT module is in the OPERATION ACTIVE mode (the RSENTnMST.OMS bits are $101_B$) and SPC communication is enabled (RSENTnCC.SPCE is $1_B$).

It is important to note that two consecutive write access might not cause a no response error as the previous request might not have started yet.

After writing to this register, the CPU should wait for at least one SPC trigger tick before writing again to this register.

### 22.3.8 RSENTnMST — RSENT Mode Status Register

**Access:** This register is read-only in 32-bit units.

**Address:** <RSENTn_base> + $0024_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | OMS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 22.16 RSENTnMST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is read. |
| 2 to 0 | OMS[2:0] | Operation Mode Status<br>000: RESET<br>001: CONFIGURATION<br>011: OPERATION IDLE<br>101: OPERATION ACTIVE<br>Other than above: Reserved |

#### RSENTnMST.OMS (Operation Mode Status)

These bits indicate the current operation mode.

These bits are read only.

These bits are updated after a mode change request is made in the RSENTnMDC.OMC register.

- 000: RESET mode

  When in RESET mode, all registers are set to their reset values and write access to all registers except the RSENTnMDC register is disabled.
  When in RESET mode, RSENT communication is disabled.

- 001: CONFIGURATION mode

  When in CONFIGURATION mode, write access to the timestamp registers (RSENTnTSPC and RSENTnTSC register), configuration registers (RSENTnCC and RSENTnBRP register), RSENTnIDE register, and mode control register (RSENTnMDC.OMC) is enabled.
  When in CONFIGURATION mode, RSENT communication is disabled.
  When entering CONFIGURATION mode, all status registers and receive buffer registers are set to their reset values.

- 011: OPERATION IDLE mode

  In OPERATION IDLE mode, no reception or SPC trigger transmission is possible.
  When entering OPERATION IDLE mode, frames in the receive buffer can be analyzed as in OPERATION ACTIVE mode, but no new frames are received.

- 101: OPERATION ACTIVE mode

  In OPERATION ACTIVE mode, reception and SPC trigger transmission are possible.

- Other than above: Reserved

### 22.3.9 RSENTnCS — RSENT Communication Status Register

**Access:** This register is read-only in 32-bit units.

**Address:** <RSENTn_base> + $0028_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SES | SMS | SCS | NRS | CVS | CLS | FNS | FES | FMS | FCS | FRS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 22.17 RSENTnCS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is read. |
| 10 | SES | Slow channel Encoding Error Interrupt Status<br>0: Not detected<br>1: Detected |
| 9 | SMS | Slow Channel Message Lost Interrupt Status<br>0: Not detected<br>1: Detected |
| 8 | SCS | Slow Channel CRC Error Interrupt Status<br>0: Not detected<br>1: Detected |
| 7 | NRS | No Response Error Interrupt Status<br>0: Not detected<br>1: Detected |
| 6 | CVS | Calibration Pulse Length Variation Error Interrupt Status<br>0: Not detected<br>1: Detected |
| 5 | CLS | Calibration Pulse Length Error Interrupt Status<br>0: Not detected<br>1: Detected |
| 4 | FNS | Fast Channel Nibble Count Error Interrupt Status<br>0: Not detected<br>1: Detected |
| 3 | FES | Fast Channel Nibble Encoding Error Interrupt Status<br>0: Not detected<br>1: Detected |
| 2 | FMS | Fast Channel Message Lost Interrupt Status<br>0: Not detected<br>1: Detected |
| 1 | FCS | Fast Channel CRC Error Interrupt Status<br>0: Not detected<br>1: Detected |
| 0 | FRS | Fast Channel Receive Interrupt Status<br>0: Not detected<br>1: Detected |

#### RSENTnCS.SES (Slow Channel Encoding Error Status)

This bit represents the slow channel encoding error status.

This bit is read only.

In the short serial message format (RSENTnCC.SMF = 01), this bit is set when the sequence on serial start bit (bit #3) is different from “1000000000000000” (a single 1 and 15 0s).

In the enhanced serial message format (RSENTnCC.SMF = 10), this bit is set when following the reception of a start sequence (“01111110”) on the serial data bit 3, bit 13 or bit 18 are not received as ‘0’.

This bit is cleared when writing 1 to RSENTnCSC.SEC.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

If the set condition occurs simultaneously with the clear condition, the bit is set.

#### RSENTnCS.SMS (Slow Channel Message Lost Status)

This bit represents the slow channel message lost status.

This bit is read only.

This bit is set when there is an attempt to update the slow channel message buffer, but the previous message has not been read yet.

This bit is cleared when writing 1 to RSENTnCSC.SMC.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

If the set condition occurs simultaneously with the clear condition, the bit is set.

#### RSENTnCS.SCS (Slow Channel CRC Error Status)

This bit represents the slow channel CRC error status.

This bit is read only.

This bit is set when a CRC error is detected on the slow channel and the slow channel CRC detection is enabled (RSENTnCC.SCCD is set to 0).

This bit is cleared when writing 1 to RSENTnCSC.SCC.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

If the set condition occurs simultaneously with the clear condition, the bit is set.

#### RSENTnCS.NRS (No Response Error Status)

This bit represents the no response error status.

This bit is read only.

This bit is set when

- the CPU writes to the RSENTnSPCT.TLL bits and
- SPC mode enabled (RSENTnCC.SPCE set to 1) and
- no complete response was received from the sensor for the previous SPC trigger.

This bit is cleared when writing 1 to RSENTnCSC.NRC.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

If the set condition occurs simultaneously with the clear condition, the bit is set.

#### RSENTnCS.CVS (Calibration Pulse Length Variation Error Status)

This bit represents the calibration pulse length variation error status.

This bit is read only.

When RSENTnCC.PPTC is 0, then this bit is set when two successive calibration pulses differ by more than 1.5625%.

When RSENTnCC.PPTC is 1, this bit is never set. In this mode (pause pulse with fixed message length), the CPU needs to check the variation of the ratio of calibration pulse to message length by reading the RSENTnCPL and RSENTnML registers.

This bit is cleared when writing 1 to RSENTnCSC.CVC.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

If the set condition occurs simultaneously with the clear condition, the bit is set.

#### RSENTnCS.CLS (Calibration Pulse Length Error Status)

This bit represents the calibration pulse length error status.

This bit is read only.

This bit is set when the measured calibration pulse length is less than 42 clock ticks or more than 70 clock ticks (deviation of 25% from nominal length).

This bit is cleared when writing 1 to RSENTnCSC.CLC.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

If the set condition occurs simultaneously with the clear condition, the bit is set.

#### RSENTnCS.FNS (Fast Channel Nibble Count Error Status)

This bit represents the fast channel nibble count error status.

This bit is read only.

This bit is set when there is an unexpected number of falling edges after the detection of a calibration pulse or between two calibration pulses.

This bit is cleared when writing 1 to RSENTnCSC.FNC.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

If the set condition occurs simultaneously with the clear condition, the bit is set.

#### RSENTnCS.FES (Fast Channel Nibble Encoding Error Status)

This bit represents the fast channel nibble encoding error status.

This bit is read only.

This bit is set when on the fast channel a measured nibble period is less than 12 clock ticks or more than 27 clock ticks.

This bit is cleared when writing 1 to RSENTnCSC.FEC.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

If the set condition occurs simultaneously with the clear condition, the bit is set.

#### RSENTnCS.FMS (Fast Channel Message Lost Status)

This bit represents the fast channel message lost status.

This bit is read only.

This bit is set when the fast channel message buffer is updated, but the previous messages in the foreground and background buffer have not been read yet.

This bit is cleared when writing 1 to RSENTnCSC.FMC.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

If the set condition occurs simultaneously with the clear condition, the bit is set.

#### RSENTnCS.FCS (Fast Channel CRC Error Status)

This bit represents the fast channel CRC error status.

This bit is read only.

This bit is set when a CRC error is detected on the fast channel and the fast channel CRC detection is enabled (RSENTnCC.FCCD is set to 0).

This bit is cleared when writing 1 to RSENTnCSC.FCC.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

If the set condition occurs simultaneously with the clear condition, the bit is set.

#### RSENTnCS.FRS (Fast Channel Receive Status)

This bit represents the fast channel receive status.

This bit is read only.

This bit is set when the fast channel receive message buffer was updated.

This bit is cleared when the CPU reads the RSENTnFRXD.FND bit.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

If the set condition occurs simultaneously with the clear condition, the bit is set.

### 22.3.10 RSENTnCSC — RSENT Communication Status Clear Register

**Access:** This register can be written in 32-bit units.

**Address:** <RSENTn_base> + $002C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SEC | SMC | SCC | NRC | CVC | CLC | FNC | FEC | FMC | FCC | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R |

**Table 22.18 RSENTnCSC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 10 | SEC | Slow Channel Encoding Error Clear<br>0: —<br>1: Clear |
| 9 | SMC | Slow Channel Message Lost Clear<br>0:—<br>1: Clear |
| 8 | SCC | Slow Channel CRC Error Clear<br>0: —<br>1: Clear |
| 7 | NRC | No Response Error Clear<br>0: —<br>1: Clear |
| 6 | CVC | Calibration Pulse Length Variation Error Clear<br>0: —<br>1: Clear |
| 5 | CLC | Calibration Pulse Length Error Clear<br>0: —<br>1: Clear |
| 4 | FNC | Fast Channel Nibble Count Error Clear<br>0: —<br>1: Clear |
| 3 | FEC | Fast Channel Nibble Encoding Error Clear<br>0: —<br>1: Clear |
| 2 | FMC | Fast Channel Message Lost Clear<br>0: —<br>1: Clear |
| 1 | FCC | Fast Channel CRC Error Clear<br>0: —<br>1: Clear |
| 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

**RSENTnCSC.SEC (Slow channel Encoding error Clear)**

Writing 1 sets RSENTnCS.SES to 0.

Writing 0 has no effect.

This bit is always read as 0.

**RSENTnCSC.SMC (Slow Channel Message Lost Clear)**

Writing 1 sets RSENTnCS.SMS to 0.

Writing 0 has no effect.

This bit is always read as 0.

**RSENTnCSC.SCC (Slow Channel CRC Error Clear)**

Writing 1 sets RSENTnCS.SCS to 0.

Writing 0 has no effect.

This bit is always read as 0.

**RSENTnCSC.NRC (No Response Error Clear)**

Writing 1 sets RSENTnCS.NRS to 0.

Writing 0 has no effect.

This bit is always read as 0.

**RSENTnCSC.CVC (Calibration Pulse Length Variation Error Clear)**

Writing 1 sets RSENTnCS.CVS to 0.

Writing 0 has no effect.

This bit is always read as 0.

**RSENTnCSC.CLC (Calibration Pulse Length Error Clear)**

Writing 1 sets RSENTnCS.CLS to 0.

Writing 0 has no effect.

This bit is always read as 0.

**RSENTnCSC.FNC (Fast Channel Nibble Count Error Clear)**

Writing 1 sets RSENTnCS.FNS to 0.

Writing 0 has no effect.

This bit is always read as 0.

**RSENTnCSC.FEC (Fast Channel Nibble Encoding Error Clear)**

Writing 1 sets RSENTnCS.FES to 0.

Writing 0 has no effect.

This bit is always read as 0.

**RSENTnCSC.FMC (Fast Channel Message Lost Clear)**

Writing 1 sets RSENTnCS.FMS to 0.

Writing 0 has no effect.

This bit is always read as 0.

**RSENTnCSC.FCC (Fast Channel CRC Error Clear)**

Writing 1 sets RSENTnCS.FCS to 0.

Writing 0 has no effect.

This bit is always read as 0.

## 22.3.11 RSENTnSRTS — RSENT Slow Channel Receive Timestamp Register

**Access:** This register is read-only in 32-bit units.

**Address:** <RSENTn_base> + $0030_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STS[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 22.19 RSENTnSRTS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | STS | Slow Channel Receive Timestamp |

### RSENTnSRTS.STS (Slow Channel Timestamp)

These bits are read only.

These bits are updated when the slow channel receive buffer is updated with the timestamp counter value of the last frame provided to the slow channel message.

These bits are cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

## 22.3.12 RSENTnSRXD — RSENT Slow Channel Receive Data Register

**Access:** This register is read-only in 32-bit units.

**Address:** <RSENTn_base> + $0034_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SND | — | SCRC[5:0] | | | | | | — | — | — | SMGC | IDD[19:16] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IDD[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 22.20 RSENTnSRXD register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | SND | Slow Channel New Data<br>0: Slow channel frame data is not updated since last read.<br>1: Slow channel frame data is updated since last read. |
| 30 | Reserved | When read, the value after reset is read. |
| 29 to 24 | SCRC[5:0] | Slow Channel CRC Data |
| 23 to 21 | Reserved | When read, the value after reset is read. |
| 20 | SMGC | Slow Channel Message Configuration<br>0: —<br>1: The slow channel receive message buffer is updated. |
| 19 to 0 | IDD[19:0] | Slow Channel Data / ID Information |

### RSENTnSRXD.SND (Slow Channel New Data)

This bit indicates that the slow channel message buffer is holding data that has not been read.

This bit is read only.

This bit is set when the slow channel receive message buffer is updated.

This bit is cleared automatically whenever it is read.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

### RSENTnSRXD.SCRC (Slow Channel CRC)

These bits are representing the slow channel CRC data.

These bits are read only.

These bits are updated when the slow channel receive message buffer is updated.

These bits are cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

### RSENTnSRXD.SMGC (Slow Channel Message Configuration)

This bit represents the slow channel message configuration bit.

This bit is read only.

This bit is updated when the slow channel receive message buffer is updated.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

**RSENTnSRXD.IDD (ID/ Data)**

These bits are representing the slow channel data and ID information.

The alignment within this register depends on the message format. For details, see **Section 22.4.3.2(5), Slow Channel Message Reception**.

These bits are read only.

These bits are updated when the slow channel receive message buffer is updated.

These bits are cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

## 22.3.13 RSENTnCPL — RSENT Calibration Pulse Length Register

**Access:** This register is read-only in 32-bit units.

**Address:** <RSENTn_base> + $0038_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CPLV16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CPLV[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 22.21 RSENTnCPL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | When read, the value after reset is read. |
| 16 to 0 | CPLV[16:0] | Calibration Pulse Length Value of Received Message |

### RSENTnCPL.CPLV (Calibration Pulse Length Value)

These bits are used by the CPU to calculate the ratio of two consecutive calibration pulses or the calibration pulse to message length in pause pulse with fixed message length mode for message diagnostics.

These bits are read only.

These bits are updated with the measured calibration pulse length in sample clock ticks when the fast channel receive message buffer is updated.

In modes other than pause pulse with fixed message length (CC.PPTC = 1'b1) or SPC mode (CC.SPCE = 1'b1), these bits are invalid and should not be used.

These bits are cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

## 22.3.14 RSENTnML — RSENT Message Length Register

**Access:** This register is read-only in 32-bit units.

**Address:** <RSENTn_base> + $003C_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | MLV[20:16] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MLV[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 22.22 RSENTnML register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. |
| 20 to 0 | MLV[20:0] | Message Length Value of Received Message |

### RSENTnML.MLV (Message Length Value)

These bits are used by the CPU to calculate the ratio of the calibration pulse to message length in pause pulse with fixed message length mode for message diagnostics.

These bits are read only.

These bits are updated with the measured message length in sample clock ticks when the fast channel receive message buffer is updated.

These bits are cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

In modes other than pause pulse with fixed message length, these bits are invalid and should not be used.

## 22.3.15 RSENTnFRTS — RSENT Fast Channel Receive Timestamp Register

**Access:** This register is read-only in 32-bit units.

**Address:** <RSENTn_base> + $0040_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FTS[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FTS[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 22.23 RSENTnFRTS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | FTS[31:0] | Fast Channel Receive Timestamp |

### RSENTnFRTS.FTS (Fast Channel Timestamp)

These bits are read only.

These bits are updated when the fast channel receive message buffer is updated.

These bits are cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

## 22.3.16 RSENTnFRXD — RSENT Fast Channel Receive Data Register

**Access:** This register is read-only in 32-bit units.

**Address:** <RSENTn_base> + $0044_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SNDM | FND | FCCN[1:0] | | FCRC[3:0] | | | | ND[23:16] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 22.24 RSENTnFRXD register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | SNDM | Slow Channel New Data Mirror<br>0: Slow channel frame data is not updated since last read.<br>1: Slow channel frame data is updated since last read. |
| 30 | FND | Fast Channel New Data<br>0: Fast channel frame data is not updated since last read.<br>1: Fast channel frame data is updated since last read. |
| 29, 28 | FCCN[1:0] | Fast Channel Communication Nibble |
| 27 to 24 | FCRC[3:0] | Fast Channel CRC Data |
| 23 to 0 | ND[23:0] | Fast Channel Nibble Data |

### RSENTnFRXD.SNDM (Slow Channel New Data Mirror)

This bit indicates that the slow channel message buffer is holding data that has not been read.

This bit is read only.

This bit is set when the slow channel receive message buffer is updated.

This bit is cleared automatically whenever the slow channel new data bit (RSENTnSRXD.SND) is read.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

### RSENTnFRXD.FND (Fast Channel New Data)

This bit indicates that the fast channel message buffer is holding data that has not been read.

This bit is read only.

This bit is set when the fast channel receive message buffer is updated.

This bit is cleared automatically whenever it is read.

This bit is cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

**RSENTnFRXD.FCCN (Fast Channel Communication Nibble)**

These bits are representing the fast channel communication nibble bits [1:0].

These bits are read only.

These bits are updated when the fast channel receive message buffer is updated.

These bits are cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

**RSENTnFRXD.FCRC (Fast Channel CRC)**

These bits are representing the fast channel CRC data.

These bits are read only.

These bits are updated when the fast channel receive message buffer is updated.

These bits are cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

**RSENTnFRXD.ND (Nibble Data)**

These bits are representing the fast channel nibble data.

The alignment of the nibble data depends on nibble count. For details, see **Section 22.4.3.2(3), Fast Channel Message Reception**.

These bits are read only.

These bits are updated when the fast channel receive message buffer is updated.

These bits are cleared when the RSENTnMST.OMS bits are changed to 001 (CONFIGURATION).

### 22.3.17 RSENT0TSSEL — RSENT0 Timestamp Mode Selection Register

This register controls the master channel of the RSENT module.

Writing to this register is only allowed when the RSENT module is in the CONFIGURATION mode (RSENTMST.OMS[2:0] = 001), or in the RESET mode (RSENTMST.OMS[2:0] = 000).

**Access:** This register can be read/written in 32-bit units.

**Address:** FFCD CF00$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSSEL 03 | MSSEL 02 | MSSEL 01 | MSSEL 00 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 22.25 RSENT0TSSEL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | MSSEL0[3:0] | Master Selection for RSENT0<br>0000: No master or RSENT0 is master.<br>0001: No master or RSENT0 is master.<br>0010: RSENT1 is the master of RSENT0.<br>0011: RSENT2 is the master of RSENT0.<br>0100: RSENT3 is the master of RSENT0.<br>0101: RSENT4 is the master of RSENT0.<br>0110: RSENT5 is the master of RSENT0.<br>0111: RSENT6 is the master of RSENT0.<br>1000: RSENT7 is the master of RSENT0.<br>1001: RSENT8 is the master of RSENT0.<br>1010: RSENT9 is the master of RSENT0.<br>Other than above: Setting prohibited |

### 22.3.18 RSENT1TSSEL — RSENT1 Timestamp Mode Selection Register

This register controls the master channel of the RSENT module.

Writing to this register is only allowed when the RSENT module is in the CONFIGURATION mode (RSENTMST.OMS[2:0] = 001), or in the RESET mode (RSENTMST.OMS[2:0] = 000).

**Access:** This register can be read/written in 32-bit units.

**Address:** FFCD CF04$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSSEL 13 | MSSEL 12 | MSSEL 11 | MSSEL 10 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 22.26 RSENT1TSSEL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | MSSEL1[3:0] | Master Selection for RSENT1<br>0000: No master or RSENT1 is master.<br>0001: RSENT0 is the master of RSENT1.<br>0010: No master or RSENT1 is master.<br>0011: RSENT2 is the master of RSENT1.<br>0100: RSENT3 is the master of RSENT1.<br>0101: RSENT4 is the master of RSENT1.<br>0110: RSENT5 is the master of RSENT1.<br>0111: RSENT6 is the master of RSENT1.<br>1000: RSENT7 is the master of RSENT1.<br>1001: RSENT8 is the master of RSENT1.<br>1010: RSENT9 is the master of RSENT1.<br>Other than above: Setting prohibited |

### 22.3.19 RSENT2TSSEL — RSENT2 Timestamp Mode Selection Register

This register controls the master channel of the RSENT module.

Writing to this register is only allowed when the RSENT module is in the CONFIGURATION mode (RSENTMST.OMS[2:0] = 001), or in the RESET mode (RSENTMST.OMS[2:0] = 000).

**Access:** This register can be read/written in 32-bit units.

**Address:** FFCD CF08$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSSEL 23 | MSSEL 22 | MSSEL 21 | MSSEL 20 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 22.27 RSENT2TSSEL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | MSSEL2[3:0] | Master Selection for RSENT2<br>0000: No master or RSENT2 is master.<br>0001: RSENT0 is the master of RSENT2.<br>0010: RSENT1 is the master of RSENT2.<br>0011: No master or RSENT2 is master.<br>0100: RSENT3 is the master of RSENT2.<br>0101: RSENT4 is the master of RSENT2.<br>0110: RSENT5 is the master of RSENT2.<br>0111: RSENT6 is the master of RSENT2.<br>1000: RSENT7 is the master of RSENT2.<br>1001: RSENT8 is the master of RSENT2.<br>1010: RSENT9 is the master of RSENT2<br>Other than above: Setting prohibited |

### 22.3.20 RSENT3TSSEL — RSENT3 Timestamp Mode Selection Register

This register controls the master channel of the RSENT module.

Writing to this register is only allowed when the RSENT module is in the CONFIGURATION mode (RSENTMST.OMS[2:0] = 001), or in the RESET mode (RSENTMST.OMS[2:0] = 000).

**Access:** This register can be read/written in 32-bit units.

**Address:** FFCD CF0C$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSSEL 33 | MSSEL 32 | MSSEL 31 | MSSEL 30 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 22.28 RSENT3TSSEL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | MSSEL3[3:0] | Master Selection for RSENT3<br>0000: No master or RSENT3 is master.<br>0001: RSENT0 is the master of RSENT3.<br>0010: RSENT1 is the master of RSENT3.<br>0011: RSENT2 is the master of RSENT3.<br>0100: No master or RSENT3 is master.<br>0101: RSENT4 is the master of RSENT3.<br>0110: RSENT5 is the master of RSENT3.<br>0111: RSENT6 is the master of RSENT3.<br>1000: RSENT7 is the master of RSENT3.<br>1001: RSENT8 is the master of RSENT3.<br>1010: RSENT9 is the master of RSENT3.<br>Other than above: Setting prohibited |

### 22.3.21 RSENT4TSSEL — RSENT4 Timestamp Mode Selection Register

This register controls the master channel of the RSENT module.

Writing to this register is only allowed when the RSENT module is in the CONFIGURATION mode (RSENTMST.OMS[2:0] = 001), or in the RESET mode (RSENTMST.OMS[2:0] = 000).

**Access:** This register can be read/written in 32-bit units.

**Address:** FFCD CF10$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSSEL 43 | MSSEL 42 | MSSEL 41 | MSSEL 40 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 22.29 RSENT4TSSEL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | MSSEL4[3:0] | Master Selection for SENT4<br>0000: No master or RSENT4 is master.<br>0001: RSENT0 is the master of RSENT4.<br>0010: RSENT1 is the master of RSENT4.<br>0011: RSENT2 is the master of RSENT4.<br>0100: RSENT3 is the master of RSENT4.<br>0101: No master or RSENT4 is master.<br>0110: RSENT5 is the master of RSENT4.<br>0111: RSENT6 is the master of RSENT4.<br>1000: RSENT7 is the master of RSENT4.<br>1001: RSENT8 is the master of RSENT4.<br>1010: RSENT9 is the master of RSENT4.<br>Other than above: Setting prohibited |

## 22.3.22 RSENT5TSSEL — RSENT5 Timestamp Mode Selection Register

This register controls the master channel of the RSENT module.

Writing to this register is only allowed when the RSENT module is in the CONFIGURATION mode (RSENTMST.OMS[2:0] = 001), or in the RESET mode (RSENTMST.OMS[2:0] = 000).

**Access:** This register can be read/written in 32-bit units.

**Address:** FFCD CF14$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSSEL 53 | MSSEL 52 | MSSEL 51 | MSSEL 50 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 22.30 RSENT5TSSEL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | MSSEL5[3:0] | Master Selection for RSENT5<br>0000: No master or RSENT5 is master.<br>0001: RSENT0 is the master of RSENT5.<br>0010: RSENT1 is the master of RSENT5.<br>0011: RSENT2 is the master of RSENT5.<br>0100: RSENT3 is the master of RSENT5.<br>0101: RSENT4 is the master of RSENT5.<br>0110: No master or RSENT5 is master.<br>0111: RSENT6 is the master of RSENT5.<br>1000: RSENT7 is the master of RSENT5.<br>1001: RSENT8 is the master of RSENT5.<br>1010: RSENT9 is the master of RSENT5.<br>Other than above: Setting prohibited |

### 22.3.23 RSENT6TSSEL — RSENT6 Timestamp Mode Selection Register

This register controls the master channel of the RSENT module.

Writing to this register is only allowed when the RSENT module is in the CONFIGURATION mode (RSENTMST.OMS[2:0] = 001), or in the RESET mode (RSENTMST.OMS[2:0] = 000).

**Access:** This register can be read/written in 32-bit units.

**Address:** FFCD CF18$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSSEL 63 | MSSEL 62 | MSSEL 61 | MSSEL 60 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 22.31 RSENT6TSSEL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | MSSEL6[3:0] | Master Selection for RSENT6<br>0000: No master or RSENT6 is master.<br>0001: RSENT0 is the master of RSENT6.<br>0010: RSENT1 is the master of RSENT6.<br>0011: RSENT2 is the master of RSENT6.<br>0100: RSENT3 is the master of RSENT6.<br>0101: RSENT4 is the master of RSENT6.<br>0110: RSENT5 is the master of RSENT6.<br>0111: No master or RSENT6 is master.<br>1000: RSENT7 is the master of RSENT6.<br>1001: RSENT8 is the master of RSENT6.<br>1010: RSENT9 is the master of RSENT6.<br>Other than above: Setting prohibited |

### 22.3.24 RSENT7TSSEL — RSENT7 Timestamp Mode Selection Register

This register controls the master channel of the RSENT module.

Writing to this register is only allowed when the RSENT module is in the CONFIGURATION mode (RSENTMST.OMS[2:0] = 001), or in the RESET mode (RSENTMST.OMS[2:0] = 000).

**Access:** This register can be read/written in 32-bit units.

**Address:** FFCD CF1C$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSSEL 73 | MSSEL 72 | MSSEL 71 | MSSEL 70 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 22.32 RSENT7TSSEL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | MSSEL7[3:0] | Master Selection for RSENT7<br>0000: No master or RSENT7 is master.<br>0001: RSENT0 is the master of RSENT7.<br>0010: RSENT1 is the master of RSENT7.<br>0011: RSENT2 is the master of RSENT7.<br>0100: RSENT3 is the master of RSENT7.<br>0101: RSENT4 is the master of RSENT7.<br>0110: RSENT5 is the master of RSENT7.<br>0111: RSENT6 is the master of RSENT7.<br>1000: No master or RSENT7 is master.<br>1001: RSENT8 is the master of RSENT7.<br>1010: RSENT9 is the master of RSENT7.<br>Other than above: Setting prohibited |

### 22.3.25 RSENT8TSSEL — RSENT8 Timestamp Mode Selection Register

This register controls the master channel of the RSENT module.

Writing to this register is only allowed when the RSENT module is in the CONFIGURATION mode (RSENTMST.OMS[2:0] = 001), or in the RESET mode (RSENTMST.OMS[2:0] = 000).

**Access:** This register can be read/written in 32-bit units.

**Address:** FFCD CF20$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSSEL 83 | MSSEL 82 | MSSEL 81 | MSSEL 80 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 22.33 RSENT8TSSEL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | MSSEL8[3:0] | Master Selection for RSENT8<br>0000: No master or RSENT8 is master.<br>0001: RSENT0 is the master of RSENT8.<br>0010: RSENT1 is the master of RSENT8.<br>0011: RSENT2 is the master of RSENT8.<br>0100: RSENT3 is the master of RSENT8.<br>0101: RSENT4 is the master of RSENT8.<br>0110: RSENT5 is the master of RSENT8.<br>0111: RSENT6 is the master of RSENT8.<br>1000: RSENT7 is the master of RSENT8.<br>1001: No master or RSENT8 is master.<br>1010: RSENT9 is the master of RSENT8.<br>Other than above: Setting prohibited |

### 22.3.26 RSENT9TSSEL — RSENT9 Timestamp Mode Selection Register

This register controls the master channel of the RSENT module.

Writing to this register is only allowed when the RSENT module is in the CONFIGURATION mode (RSENTMST.OMS[2:0] = 001), or in the RESET mode (RSENTMST.OMS[2:0] = 000).

**Access:** This register can be read/written in 32-bit units.

**Address:** FFCD CF24$_H$

**Value after reset:** 0000 0000$_H$ This register is initialized by various types of reset.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSSEL 93 | MSSEL 92 | MSSEL 91 | MSSEL 90 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 22.34 RSENT9TSSEL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | MSSEL9[3:0] | Master Selection for RSENT9<br>0000: No master or RSENT9 is master.<br>0001: RSENT0 is the master of RSENT9.<br>0010: RSENT1 is the master of RSENT9.<br>0011: RSENT2 is the master of RSENT9.<br>0100: RSENT3 is the master of RSENT9.<br>0101: RSENT4 is the master of RSENT9.<br>0110: RSENT5 is the master of RSENT9.<br>0111: RSENT6 is the master of RSENT9.<br>1000: RSENT7 is the master of RSENT9.<br>1001: RSENT8 is the master of RSENT9.<br>1010: No master or RSENT9 is master.<br>Other than above: Setting prohibited |

## 22.4 Operation

### 22.4.1 Modes of Operation

The RSENT module can be in one of the following modes:

- RESET mode
- CONFIGURATION mode
- OPERATION IDLE mode
- OPERATION ACTIVE mode

**Figure 22.2** shows the possible transitions between the channel modes:

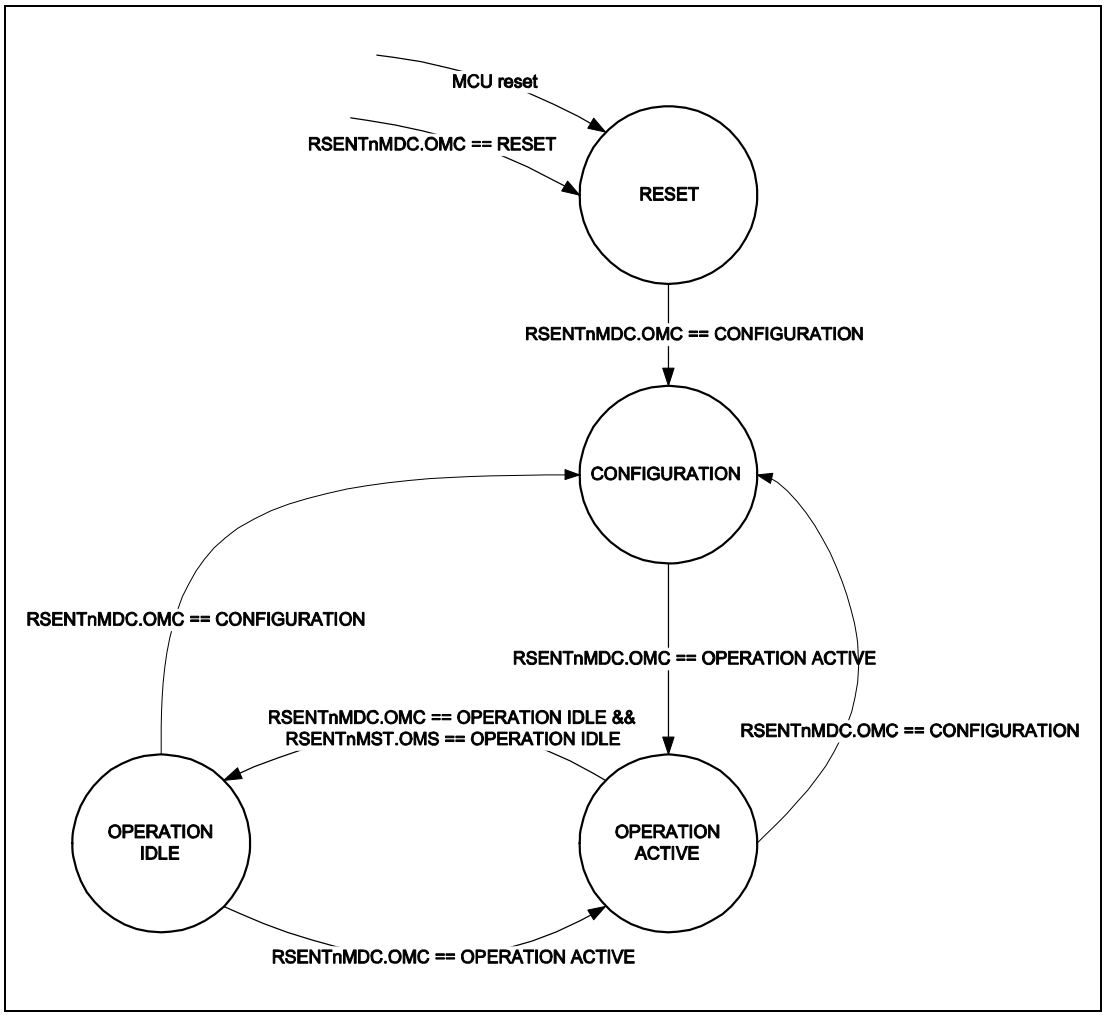



**Figure 22.2 Transition between Operation Modes**

The current operation mode status can be seen in the RSENTnMST.OMS bits.

#### 22.4.1.1 RESET Mode

This mode is the initial mode that the RSENT module automatically enters after the hardware reset (MCU reset) is cleared. Its purpose is to provide a clean reset of the registers in the RSENT module.

The RESET mode is also entered after the RSENTnMDC.OMC bits have been set to $000_B$. In this state, all, configuration, control (except RSENTnMDC.OMC bits), and status registers are set to their reset value. Any on-going transmission or reception process is stopped immediately and the interface pins of the RSENT module are set to their default values.

Read access to all registers is possible in this state. Write access is limited to the RSENTnMDC register.

#### 22.4.1.2 CONFIGURATION Mode

The CONFIGURATION mode is entered after the RSENTnMDC.OMC bits have been set to $001_B$.

Any on-going transmission or reception process is stopped immediately and the interface pins of the RSENT module are set to their default values.

In this state, all status registers (RSENTnCS) and the receive buffer registers (RSENTnSRTS, RSENTnSRXD, RSENTnCPL, RSENTnML, RSENTnFRTS, and RSENTnFRXD) are set to their default value.

Read access to all registers is possible in this state.

Write access is limited to both timestamp registers (TSCP and RSENTnTSC) and configuration registers (RSENTnCC, RSENTnBRP, RSENTnIDE, and RSENTnMDC).

#### 22.4.1.3 OPERATION IDLE Mode

This mode is entered after the RSENTnMDC.OMC bits have been set to $011_B$.

In OPERATION IDLE mode, no reception and transmission are done.

When entering OPERATION IDLE mode, frames in the receive buffer can be analyzed as in OPERATION ACTIVE mode, but no new frames are received.

Read access to all registers is possible in this state.

Write access is granted only to RSENTnTSC, RSENTnIDE, RSENTnMDC, and RSENTnCSC.

#### 22.4.1.4 OPERATION ACTIVE Mode

This mode is entered after the RSENTnMDC.OMC bits have been set to $101_B$.

In OPERATION ACTIVE mode, transmission and reception can take place.

Read access to all registers is possible in this state.

Write access is granted only to RSENTnTSC, RSENTnIDE, RSENTnMDC, RSENTnSPCT, and RSENTnCSC.

### 22.4.1.5 Register Behavior in Operation Modes

**Table 22.35** shows the register behavior when the RSENT module transitions to the indicated operation modes. The table also gives an overview about the access restriction in each operation mode.

**Table 22.35 Register Behavior in Operation Modes**

| | | MCU Reset | RESET | | CONFIGURATION | | OPERATION IDLE | | OPERATION ACTIVE | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Register Name** | **Symbol** | **Change** | **Change** | **R/W** | **Change** | **R/W** | **Change** | **R/W** | **Change** | **R/W** |
| Timestamp prescaler configuration register | RSENTnTSPC | $0000\ 0000_H$ | $0000\ 0000_H$ | R | Unchanged | R/W | Unchanged | R | Unchanged | R |
| Timestamp counter register | RSENTnTSC | $0000\ 0000_H$ | $0000\ 0000_H$ | R | Unchanged | R/W | Unchanged | R/W *1 | Unchanged | R/W *1 |
| Communication configuration register | RSENTnCC | $0000\ 0000_H$ | $0000\ 0000_H$ | R | Unchanged | R/W | Unchanged | R | Unchanged | R |
| Baud rate prescaler register | RSENTnBRP | $0000\ 0000_H$ | $0000\ 0000_H$ | R | Unchanged | R/W | Unchanged | R | Unchanged | R |
| Interrupt/DMA enable register | RSENTnIDE | $0000\ 0000_H$ | $0000\ 0000_H$ | R | Unchanged | R/W | Unchanged | R/W | Unchanged | R/W |
| Mode control register | RSENTnMDC | $0000\ 0000_H$ | $0000\ 0000_H$ | R/W | Unchanged | R/W | Unchanged | R/W | Unchanged | R/W |
| SPC transmission register | RSENTnSPCT | $0000\ 0000_H$ | $0000\ 0000_H$ | R | Unchanged | R | Unchanged | R | Unchanged | R/W |
| Mode status register | RSENTnMST | $0000\ 0000_H$ | $0000\ 0000_H$ | R | $0000\ 0001_H$ | R | $0000\ 0003_H$ | R | $0000\ 0005_H$ | R |
| Communication status register | RSENTnCS | $0000\ 0000_H$ | $0000\ 0000_H$ | R | $0000\ 0000_H$ | R | Unchanged | R | Unchanged | R |
| Communication status clear register | RSENTnCSC | $0000\ 0000_H$ | $0000\ 0000_H$ | R | Unchanged | R/W | Unchanged | R/W | Unchanged | R/W |
| Slow channel receive timestamp register | RSENTnSRTS | $0000\ 0000_H$ | $0000\ 0000_H$ | R | $0000\ 0000_H$ | R | Unchanged | R | Unchanged | R |
| Slow channel receive data register | RSENTnSRXD | $0000\ 0000_H$ | $0000\ 0000_H$ | R | $0000\ 0000_H$ | R | Unchanged | R | Unchanged | R |
| Calibration pulse length register | RSENTnCPL | $0000\ 0000_H$ | $0000\ 0000_H$ | R | $0000\ 0000_H$ | R | Unchanged | R | Unchanged | R |
| Message length register | RSENTnML | $0000\ 0000_H$ | $0000\ 0000_H$ | R | $0000\ 0000_H$ | R | Unchanged | R | Unchanged | R |
| Fast channel receive timestamp register | RSENTnFRTS | $0000\ 0000_H$ | $0000\ 0000_H$ | R | $0000\ 0000_H$ | R | Unchanged | R | Unchanged | R |
| Fast channel receive data register | RSENTnFRXD | $0000\ 0000_H$ | $0000\ 0000_H$ | R | $0000\ 0000_H$ | R | Unchanged | R | Unchanged | R |

Note 1. Means write restriction exists.

## 22.4.2 Clock Configuration

### 22.4.2.1 Timestamp

#### (1) Timestamp Clock Configuration

RSENT incorporates the timestamp counter.

The minimum required resolution of the timestamp is 1 µs. Depending on the applied communication frequency, the user should configure the RSENTnTSPC.TTPV bits to achieve the 1-µs resolution. The input frequency is divided by the configured timestamp prescaler value RSENTnTSPC.TTPV.

Depending on the configured tick lengths, the resolution can be decreased by configuring the RSENTnTSPC.TTM bits. The already divided input frequency is divided further by the value of the TSPC.TTM bits.

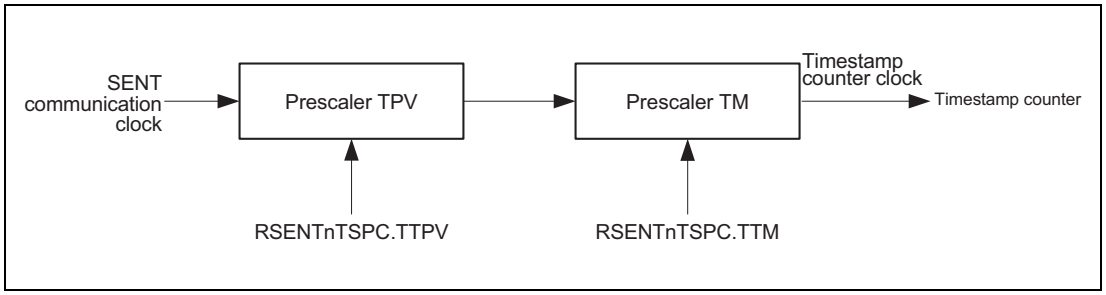



**Figure 22.3 Timestamp Counter Clock Generation**

#### (2) Timestamp Counter Operation

The timestamp counter value can be initialized to any value by writing to the RSENTnTSC.TS bits only when the RSENT module is in CONFIGURATION mode.

When timestamp counters are configured to operate in master mode (RSENTnTSPC.TMS = 0), the CPU can reset the timestamp counter by writing 0000 $0000_H$ to the RSENTnTSC.TS bits when the RSENT module is in OPERATION_IDLE or OPERATION_ACTIVE mode.

When timestamp counters are configured to operate in slave mode (RSENTnTSPC.TMS = 1), the timestamp counter is cleared when the CPU writes 0000 $0000_H$ to the RSENTnTSC.TS bits of master RSENT module when the RSENT module is in OPERATION_IDLE or OPERATION_ACTIVE mode. The RSENT module operating in slave mode should have the same timestamp counter prescaler settings as master RSENT module. When timestamp counter synchronization occurs, the internal timestamp counter prescalers are also synchronized.

The current timestamp counter value can be read from the RSENTnTSC.TS bits.

When the RSENT module is in OPERATION ACTIVE mode, each received message is stored with its related timestamp. Timestamp values are taken for fast channel and slow channel data.

The timestamp value is captured when the calibration pulse is detected.

The timestamp value for the fast channel is stored in the RSENTnFRTS.STS bits when the fast channel receive message buffer is updated.

The timestamp value for the slow channel is stored in the RSENTnSRTS.STS bits. The timestamp value for the slow channel is identical to the timestamp value of the last fast channel message contributing to the slow channel message.

In case timestamp counter synchronization is required, the following flow should be used.

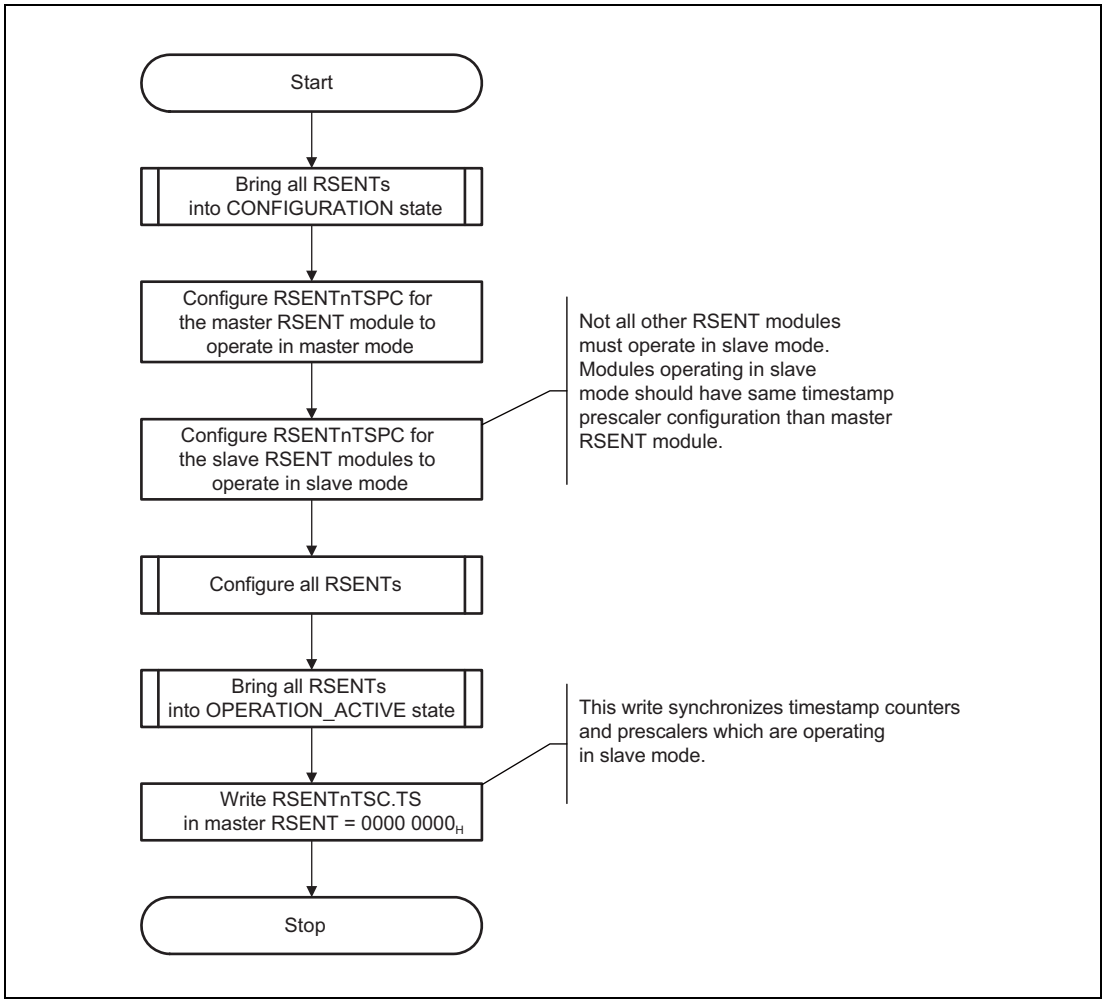
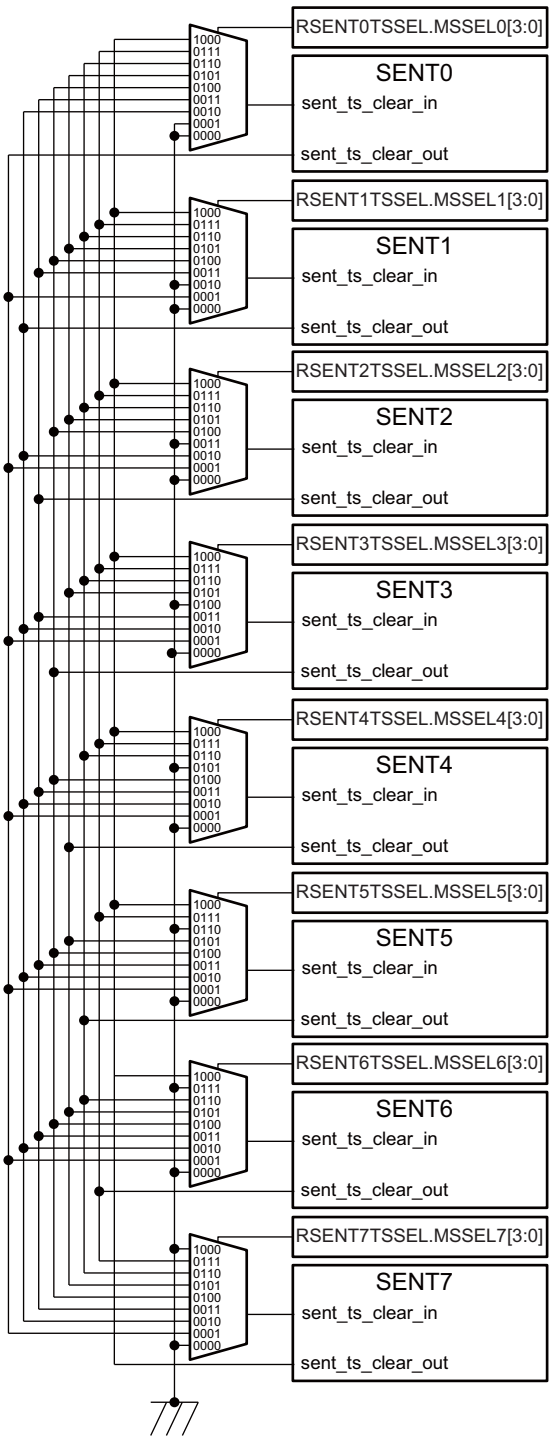



**Figure 22.4 Timestamp Counter Synchronization**

Further synchronization can be done as long as master RSENT module is in either OPERATION_ACTIVE or OPERATION_IDLE state.

In order to realize the specified timestamp counter synchronization the RSENT modules are interconnected as shown **Figure 22.5**.




**Note:** Only P1H-CE has SENT8 and SENT9.

**Figure 22.5 Timestamp Signal Connection among Channels**

### 22.4.2.2 Communication Clock Configuration

#### (1) RX BRP Setting

Depending on the applied “RSENT communication clock” frequency the RSENTnBRP.SCDV and RSENTnBRP.SCMV registers must be configured to achieve the required sample frequency ($f_{SAMPLE}$) of 16MHz.

The values should be selected according to the following formula, in such a way that the lowest terms (e.g. 2/3 instead of 4/6) is generated.

Divide the communication clock (RSENT communication clock) and select the values of RSENTnBRP.SCMV and RSENTnBRP.SCDV to achieve $f_{SAMPLE}$ of 16MHz.

fSAMPLE = 16MHz = $f_{COMUNICATION}$ × (RSENTnBRP.SCMV/RSENTnBRP.SCDV)

The communication clock is selected frequency within the range of 32MHz to 80MHz or equal to 16MHz

$$f_{SAMPLE} = 16MHz = f_{COMUNICATION} \times \frac{RSENTnBRP.SCMV}{RSENTnBRP.SCDV}$$

#### (2) RX and SPC Tick Settings

The used tick length in RX and SPC function can be configured with the RSENTnBRP.TTI and RSENTnBRP.TTF bits. Tick lengths from 1.0 µs to 90.0 µs with a resolution of 0.1 µs can be configured.

The RSENTnBRP.TTI holds the integer part of the tick length and the RSENTnBRP.TTF bits hold the fractional part of the tick length. The tick length is then calculated by:

$$T_{TICK} = T_{RSENTnBRP.TTI} + T_{RSENTnBRP.TTF}$$

**NOTE**

Set Tick lengths more than 10us when clkc frequency set less than 32MHz.

## 22.4.3 RSENT Operation

### 22.4.3.1 Changing Operation Modes

Once initialization has been completed in CONFIGURATION mode, operation can be enabled by entering OPERATION ACTIVE mode. This is done by setting the RSENTnMDC.OMC bits to OPERATION ACTIVE and waiting for the RSENTnMST.OMS to transition to OPERATION ACTIVE.

Once in OPERATION ACTIVE mode the RSENT module begins to receive messages or SPC communication can be started depending on the configuration.

**Figure 22.6** shows the communication enabled flow assuming that the RSENT module is in RESET mode:

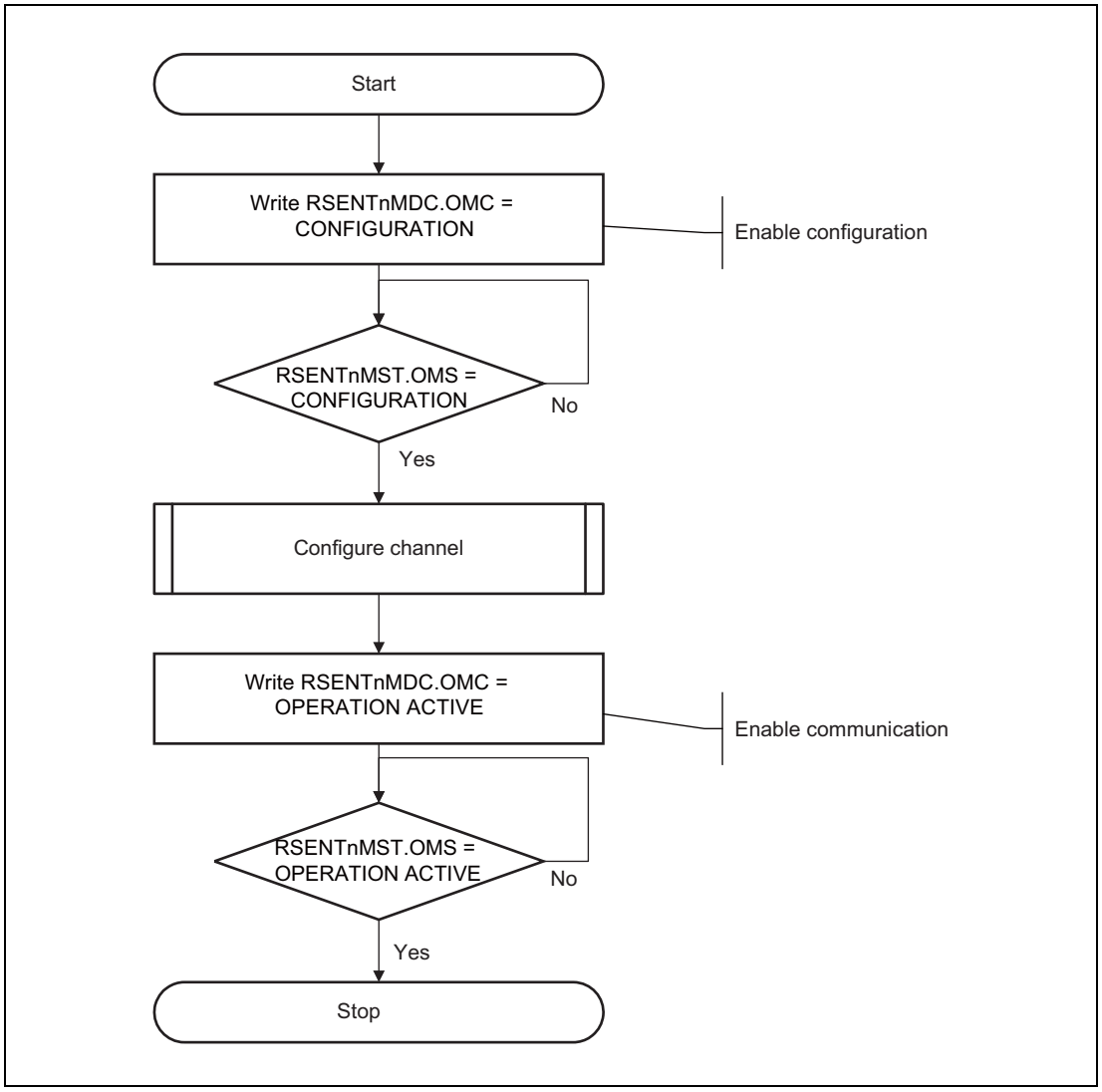



**Figure 22.6 Communication Enable Flow**

To leave OPERATION ACTIVE mode, communication should be disabled first by transitioning to OPERATION IDLE mode. This is done by setting the RSENTnMDC.OMC bits to OPERATION IDLE and waiting for the RSENTnMST.OMS bits to transition to OPERATION IDLE.

The transition between OPERATION ACTIVE and OPERATION IDLE depends on the setting of the RSENTnCC.SPCE bit.

**(1) RSENTnCC.SPCE = 0**

In case a reception is currently ongoing, the mode change from OPERATION ACTIVE to OPERATION IDLE takes place when the receive buffer was updated (see **Section 22.4.3.2(3), Fast Channel Message Reception**).

In case no reception is ongoing, the mode change from OPERATION ACTIVE to OPERATION IDLE takes place immediately.

**(2) RSENTnCC.SPCE = 1**

In case a reception is ongoing, the mode change from OPERATION ACTIVE to OPERATION IDLE takes place when the falling edge of the end pulse is received.

In case a no response error is flagged, the mode change from OPERATION ACTIVE to OPERATION IDLE takes place at the same time as the error flagging.

The mode change from OPERATION ACTIVE to OPERATION IDLE takes place when the sequence of making a SPC trigger and receiving the response has been completed. This means when a response was already received, the transition takes place immediately. When the response is still pending, the mode change from OPERATION ACTIVE to OPERATION IDLE takes place when the falling edge of the end pulse is received.

CONFIGURATION mode can be entered at any time by writing CONFIGURATION to the RSENTnMDC.OMC bits and waiting for the RSENTnMST.OMS to transition to CONFIGURATION.

Once CONFIGURATION mode is entered, the remaining status and message information stored in the RSENT module is lost since status and message information is cleared in CONFIGURATION mode.

**Figure 22.7** shows the communication disable flow assuming that the RSENT module is in OPERATION ACTIVE mode.

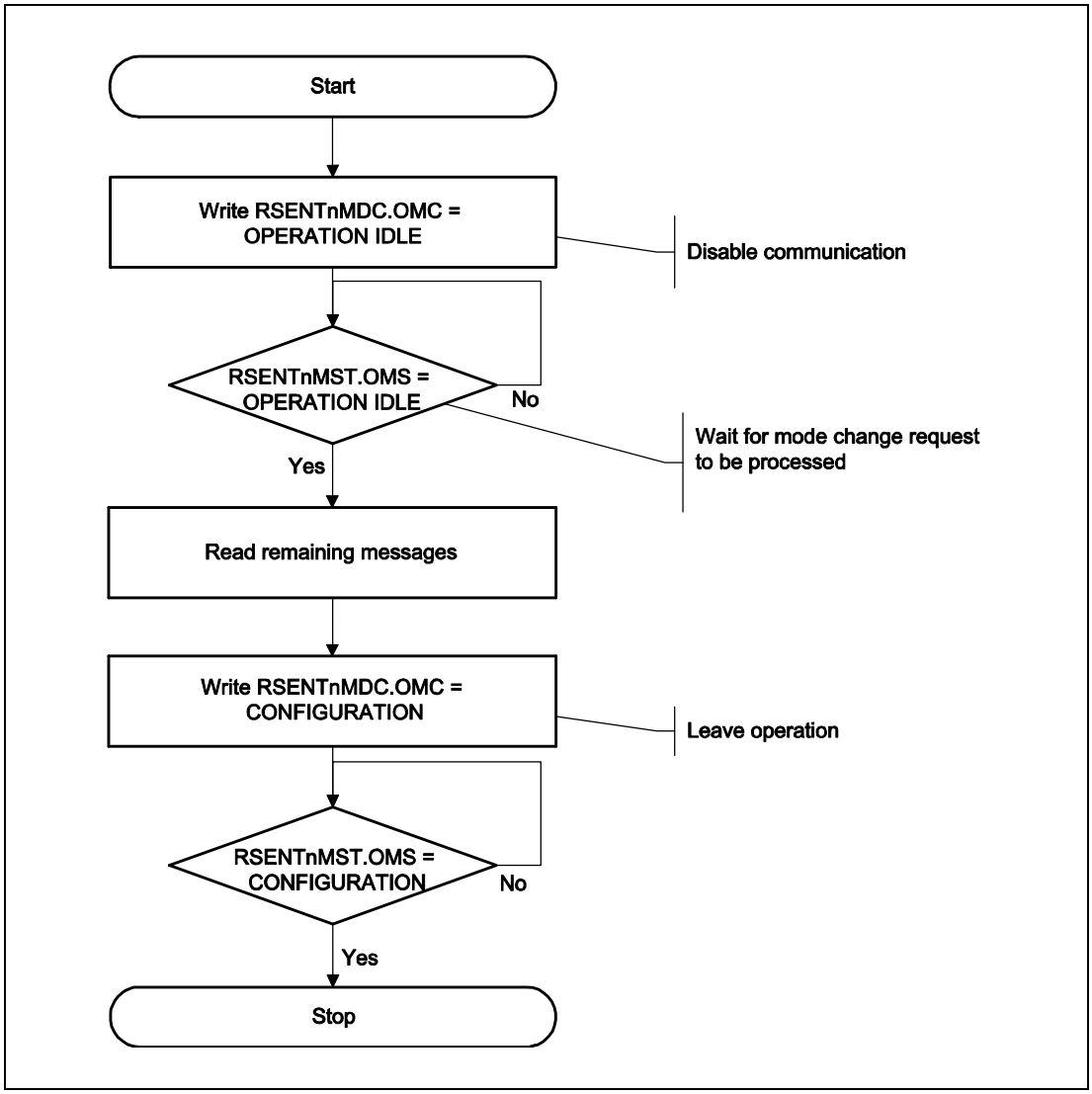



**Figure 22.7 Communication Disable Flow**

**Table 22.36 Transition time from each mode**

| Transition | Condition | Transition Time |
|---|---|---|
| RESET to CONFIGURATION | | 5 PCLK + 3 CLKC |
| CONFIGURATION to OPERATION ACTIVE | | 5 PCLK + 6 CLKC |
| OPERATION ACTIVE to OPERATION IDLE | | Falling edge of next frame sync nibble + 5 sample clocks + 4 CLKC + 4 PCLK |
| | In case frame reception has not started | 5 PCLK + 1 sample clock + 7 CLKC |
| OPERATION IDLE to CONFIGURATION | | 4 PCLK + 4 CLKC |
| OPERATION ACTIVE to CONFIGURATION | | 5 PCLK + 7 CLKC + 1 sample clock |
| OPERATION IDLE to OPERATION ACTIVE | | 5 PCLK + 6 CLKC |

### 22.4.3.2 Message Reception

RSENT message reception is composed of the calibration pulse reception followed by the data nibble pulse reception.

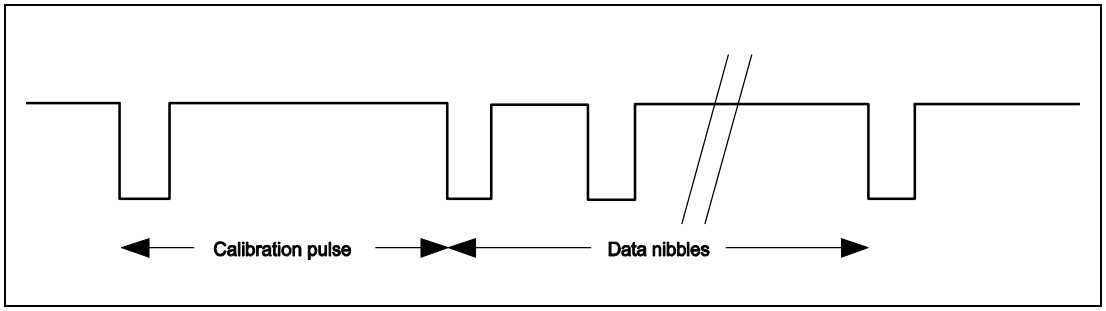



Figure 22.8 RSENT Received Message Structure

#### (1) Calibration Pulse Reception

Within the calibration pulse reception phase the internally generated clock tick is adjusted to the transmit clock speed.

In addition, the calibration pulse is used to end the previous message and perform message diagnostics. The RSENT module supports automatic calibration pulse length diagnostics in variable message length modes (RSENTnCC.PPTC = 0). In case the calibration pulse ratio check fails, the calibration pulse length variation error flag (RSENTnCS.CVS) is set to 1.

#### (2) Data Nibble Reception

The receive function of the RSENT module is a capture and compare function. The RSENT module receives sensor information encoded by the temporal distance of two consecutive falling edges on the data line. The temporal distance (in # of clock ticks) is captured and compared against a set of values to determine the actual nibble value. The data encoding is illustrated in **Table 22.37** below.

Table 22.37 Data Nibble Encoding

| Nibble Period (# Clock Ticks) | Nibble Value (Binary) |
|---|---|
| 12 | $0000_B$ |
| 13 | $0001_B$ |
| 14 | $0010_B$ |
| 15 | $0011_B$ |
| 16 | $0100_B$ |
| 17 | $0101_B$ |
| 18 | $0110_B$ |
| 19 | $0111_B$ |
| 20 | $1000_B$ |
| 21 | $1001_B$ |
| 22 | $1010_B$ |
| 23 | $1011_B$ |
| 24 | $1100_B$ |
| 25 | $1101_B$ |
| 26 | $1110_B$ |
| 27 | $1111_B$ |

The received data nibbles are composed into an RSENT message which is then stored in the fast channel receive message buffer.

Any other received nibble period during the reception of data nibbles will cause a fast channel nibble encoding error.

### (3) Fast Channel Message Reception

Messages received on the fast message channel are stored in a receive buffer.

A fast channel receive buffer is composed of the calibration pulse length register (RSENTnCPL), the message length register (RSENTnML), the fast channel receive timestamp register (RSENTnFRTS), and the fast channel receive data register (RSENTnFRXD).

All registers are placed on successive addresses that allows transferring the register content into memory using DMA.

The RSENT module is equipped with a double receive buffer structure that allows the storage of two complete RSENT messages including the related timestamp and message length information. Message decoding and assembling are done in a separate register stage.

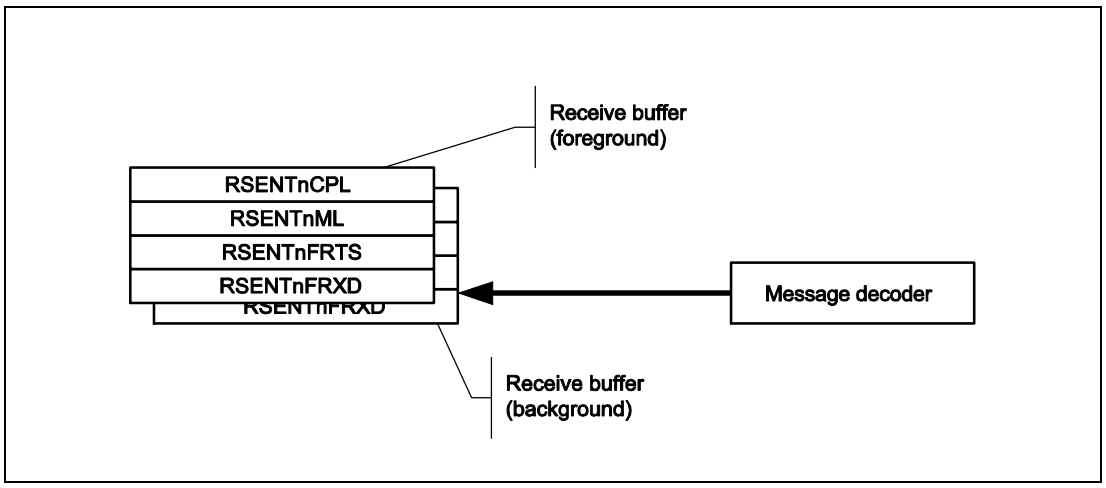



**Figure 22.9 Fast Channel Receive Message Buffer**

The first received message is placed into the message buffer that can be accessed by the CPU. This buffer (except the RSENTnFRXD.SNDM bit) is not updated any more until the RSENTnFRXD.FND bit was read.

When a new message is placed into a receive buffer, the RSENTnFRXD.FND bit is set. At the same time, the RSENTnCS.FRS bit is set and, if enabled, a receive interrupt request is generated.

When the foreground receive buffer is holding an unprocessed message (the RSENTnFRXD.FND bit is 1), any further incoming message is placed in the background buffer. The background buffer is updated with any further incoming messages. In case an unprocessed background message buffer message is overwritten, the RSENTnCS.FMS bit is set to 1.

When the CPU reads the RSENTnFRXD.FND bit and there is valid data in the background buffer, the data previously located in the background buffer becomes available in the receive buffer and is accessible by the CPU. If enabled, a new interrupt request for fast channel data is generated and RSENTnCS.FRS is set.

When the RSENTnFRXD.FND/ RSENTnCS.FRS bit is not set, the data in the receive buffer is not defined and the CPU should not access the receive buffer.

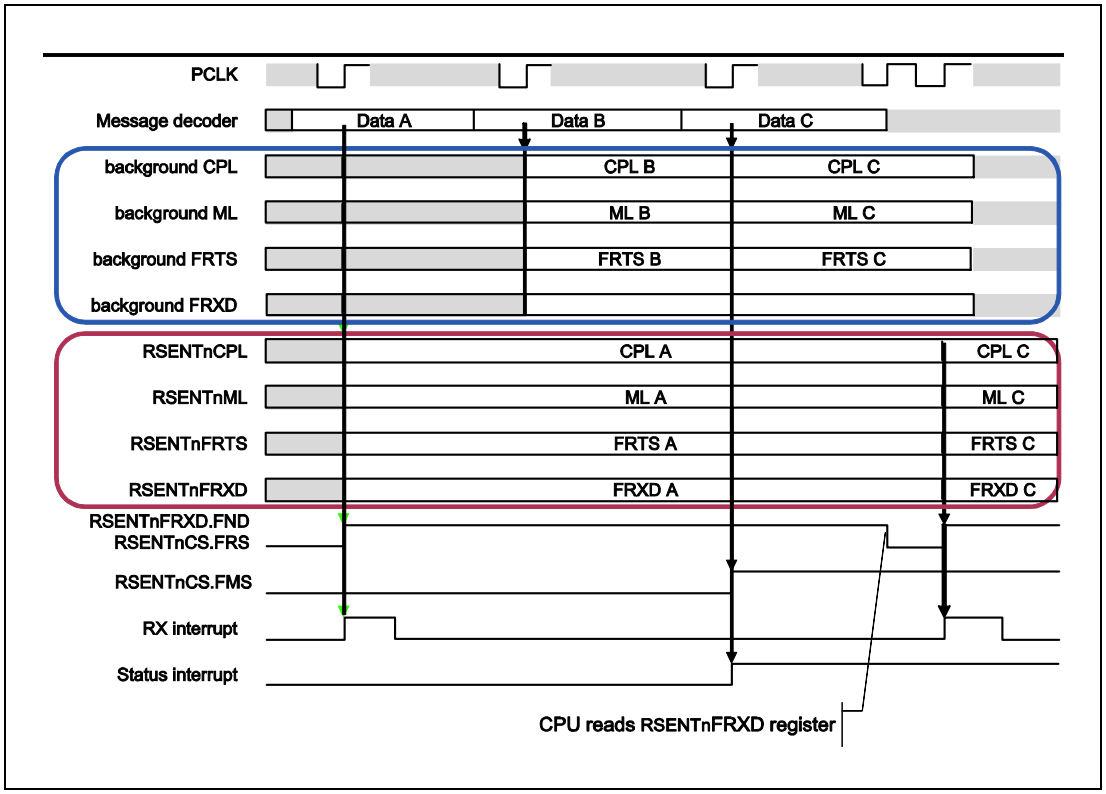



**Figure 22.10 Fast Channel Receive Buffer Update Timing**

The update timing of the receive buffer depends on the applied configuration as depicted in **Figure 22.11** to **Figure 22.14**.

The data alignment in the RSENTnFRXD register depends on the nibble data count (RSENTnCC.NDN).

**Table 22.38 Data Nibble Alignment in RSENTnFRXD Register**

| RSENTnCC.NDN | 23:20 | 19:16 | 15:12 | 11:8 | 7:4 | 3:0 |
|---|---|---|---|---|---|---|
| $000_B$ | Undefined | Undefined | Undefined | Undefined | Undefined | Nibble 1 |
| $001_B$ | Undefined | Undefined | Undefined | Undefined | Nibble 1 | Nibble 2 |
| $010_B$ | Undefined | Undefined | Undefined | Nibble 1 | Nibble 2 | Nibble 3 |
| $011_B$ | Undefined | Undefined | Nibble 1 | Nibble 2 | Nibble 3 | Nibble 4 |
| $100_B$ | Undefined | Nibble 1 | Nibble 2 | Nibble 3 | Nibble 4 | Nibble 5 |
| $101_B$ | Nibble 1 | Nibble 2 | Nibble 3 | Nibble 4 | Nibble 5 | Nibble 6 |

1. SAE operation with variable message length and preferred check method (RSENTnCC.SPCE = 0, RSENTnCC.PPTC = 0, RSENTnCC.FCM = 0)

In this operation mode, the RSENT module automatically performs the check for successive calibration pulse variation according to the preferred option in the J2716 2010 specification. In this mode, message diagnostics is done after the calibration pulse was received following a message.

If this check is passed, the receive message buffer is updated.

If this check is not passed, the receive message buffer is not updated and RSENTnCS.CVS is set to 1.

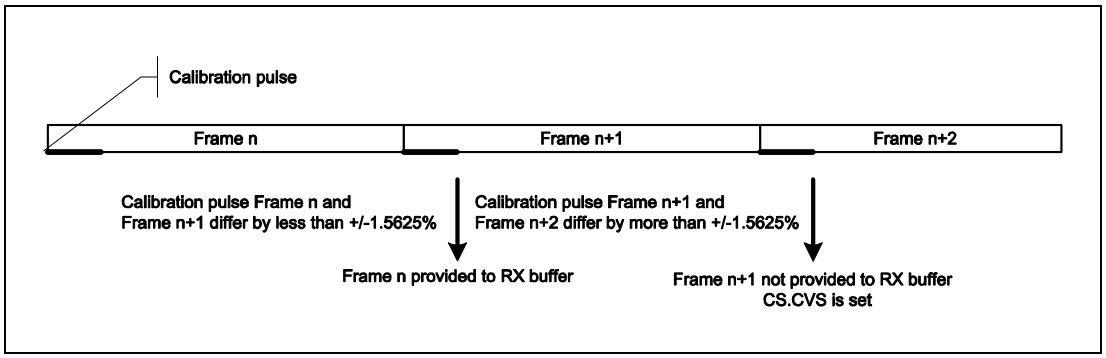



**Figure 22.11 Buffer Update in Variable Message Length Mode and Preferred Check Method**

2. SAE operation with variable message length and optional check method (RSENTnCC.SPCE = 0, RSENTnCC.PPTC = 0, RSENTnCC.FCM = 1)

In this operation mode, the RSENT module automatically performs the check for successive calibration pulse variation according to the optional frame check method as described in the J2716 2010 specification. In this mode, the calibration pulse of the current frame is compared to the calibration pulse of the last valid preceding frame.

If this check is passed, the receive message buffer is updated.

If this check is not passed, the receive message buffer is not updated and RSENTnCS.CVS is set to 1.

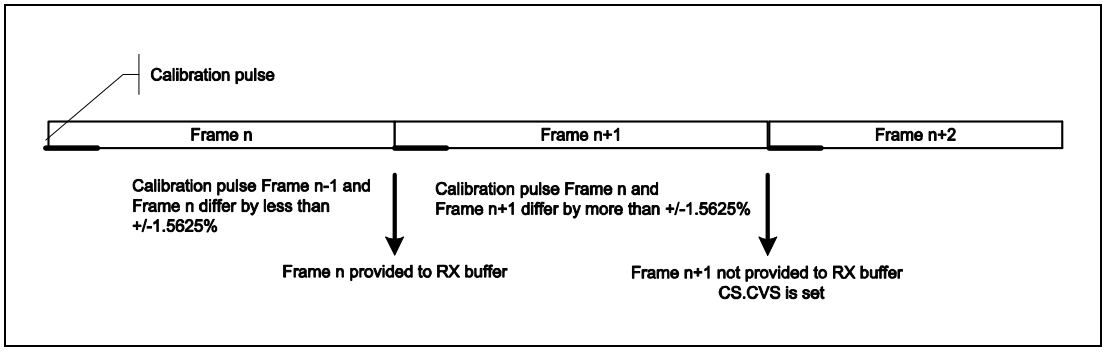



**Figure 22.12 Buffer Update in Variable Message Length Mode and Optional Check Method**

3. SAE operation with fixed message length (RSENTnCC.SPCE = 0, RSENTnCC.PPTC = 1)

In this mode, the RSENT module does not perform the check for calibration pulse and message length ratio according to the preferred option in the J2716 2010 specification. In this mode, the RSENT module provides the calibration pulse length in the RSENTnCPL register and the message length information in the RSENTnML register. The numbers provided are based on samples.

The message buffer is updated at the beginning of the following calibration pulse irrespective of the values in the RSENTnCPL and RSENTnML registers. The CPU needs to calculate the ratio and either accept or discard the message.

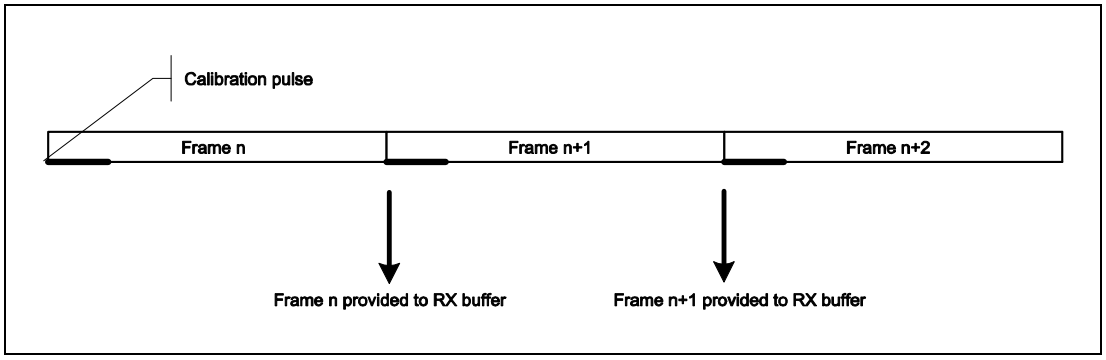



**Figure 22.13 Buffer Update in Fixed Message Length Mode**

The RSENTnCS.CVS (calibration pulse width variation error status) bit is never set in this mode.

4. SPC operation (RSENTnCC.SPCE = 1)

In this operation mode, sensor data transmission is done following a SPC master trigger pulse. Within SAE RSENT communication, the calibration pulse or pause pulse is terminating the previous message. In SPC communication, the sensor is only sending data following a SPC trigger request. An end pulse sent by the sensor is terminating the message. The message buffer is updated at the beginning of the end pulse.

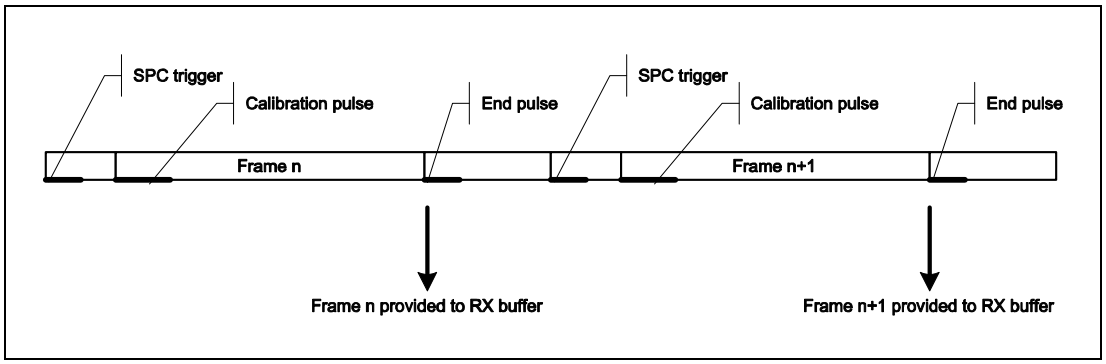



**Figure 22.14 Buffer Update in SPC Mode**

The RSENTnCS.CVS (calibration pulse width variation error status) bit is never set in this mode.

The RSENT module provides the calibration pulse length in the RSENTnCPL register and the message length information in the RSENTnML register. The numbers provided are based on samples. The CPU needs to calculate the ratio of calibration pulses and/or message length and either accept or discard the message.

In case of variable message length mode, the RSENT module cannot perform this check because the receive timing of the next calibration pulse depends on the next SPC trigger timing.

### (4) Fast Channel Reception Flow

In **Figure 22.15**, the recommended reception flow for the fast channel receive buffer is shown.

When using a polling or event driven method, the CPU should only read the setting of the RSENTnCS.FRS bit to check the availability of new fast channel data.

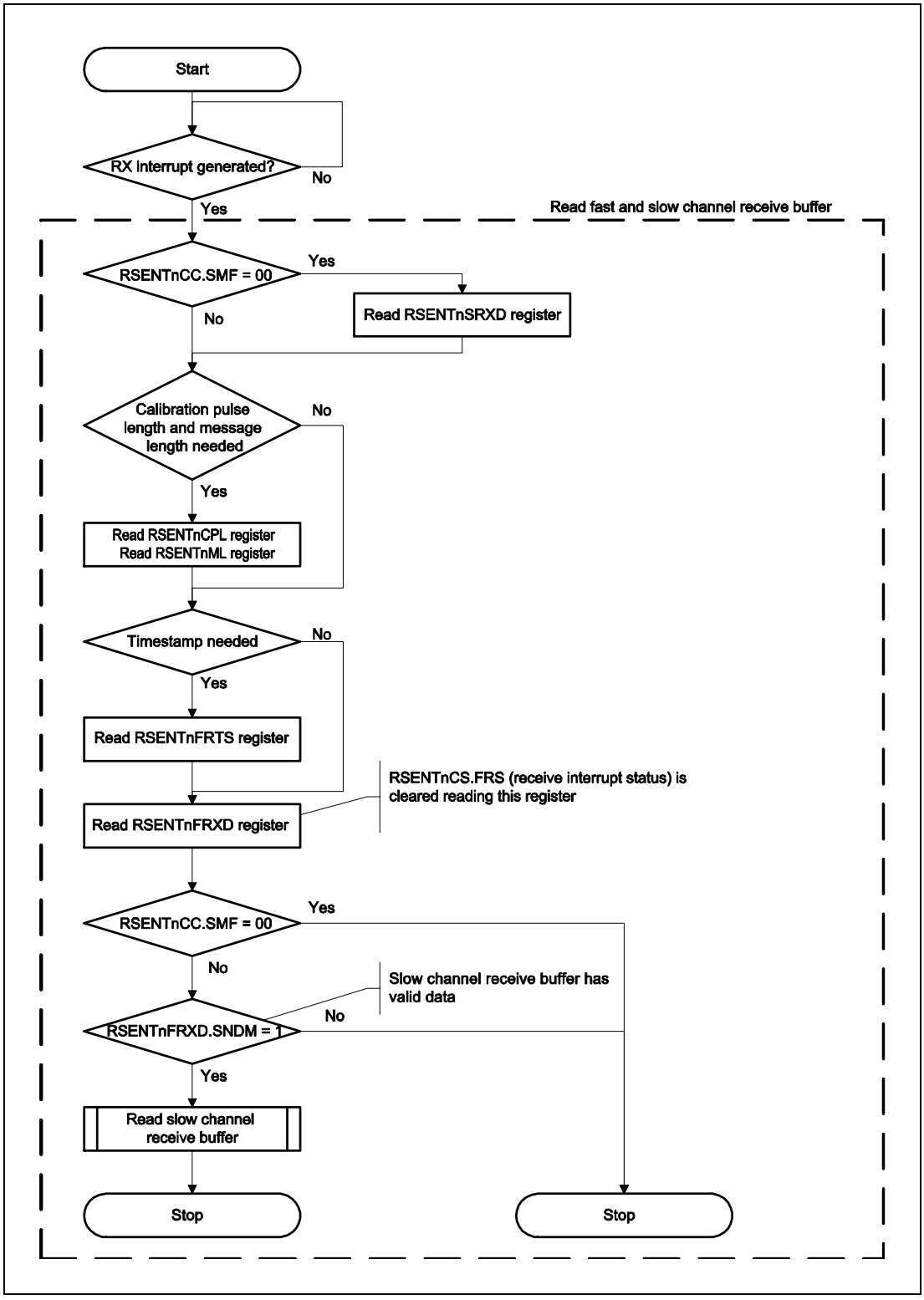



**Figure 22.15 Fast Channel Reception Flow**

In any case, the CPU should keep the order in reading the receive buffer registers as shown in the flow. The RSENTnFRXD register should be the last register to be accessed.

The handling of the slow channel receive buffer is described in **(6) Slow Channel Reception Flow**.

In case of SAE communication with pause pulse and fixed message length, the flow must be extended by checking of the ratio of the calibration pulse to message length. This variation check must be performed by the CPU. In case the variation check fails, the CPU must discard the received message.

### (5) Slow Channel Message Reception

The RSENT module supports extraction of the slow message out of the fast channel messages by using the bits 3 and 2 out of the status and communication nibble. In order to enable the slow channel extraction, the CPU should set the RSENTnCC.SMF bits to the expected serial message format.

When no serial message extraction is selected (RSENTnCC.SMF = 00), the RSENTnSRXD register becomes part of the fast channel receive buffer structure (including background buffer) and RSENTnSRTS register should be ignored. The status and communication nibble is placed in the RSENTnSRXD.IDD bits. Furthermore no slow channel new data and slow channel message lost flags are generated.

In order to receive the slow channel serial message, all fast channel serial messages contributing to a slow channel serial message must be received successfully and the received slow channel serial message must comply with the selected serial message format.

A message lost on the fast channel does not impact the reception on the slow channel.

A slow channel receive buffer is composed of the slow channel receive timestamp register (RSENTnSRTS) and the slow channel receive data register (RSENTnSRXD).

In opposite to the fast channel receive buffer, the slow channel receive buffer does not support a double receive buffer structure; only a single receive buffer structure is available.

Message decoding and assembling is done in a separate register stage.

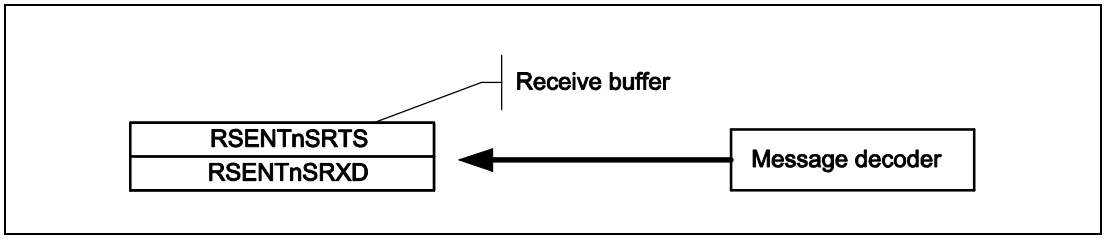



**Figure 22.16 Slow Channel Receive Message Buffer**

The slow channel receive buffer is updated at the same time as the fast channel receive buffer that holds the last status and communication nibble required for the slow channel message. The RSENTnSRXD.SND bit is set to 1 at the same time.

Further updates to the buffer are not carried out until after the RSENTnSRXD.SND bit has been read.

When the receive buffer is holding an unprocessed message (RSENTnSRXD.SND is 1), any further incoming message is lost (the slow channel receive buffer is not updated) and RSENTnCS.SMS is set to 1.

When the CPU reads the RSENTnSRXD register, RSENTnSRXD.SND is automatically cleared.

The RSENTnSRTS register is updated with the current timestamp counter register value of the last frame contributing to the slow channel message.

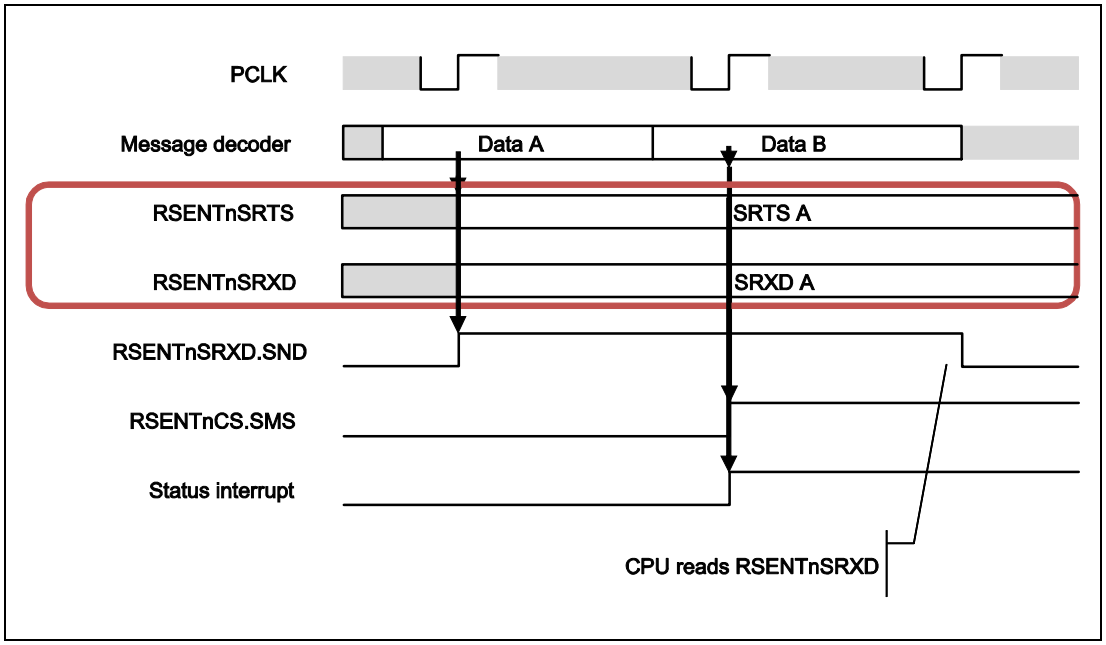



**Figure 22.17 Slow Channel Receive Buffer Update Timing**

The data alignment in the RSENTnSRXD register depends on the slow channel message format (RSENTnCC.SMF) and the received configuration bit.

**Table 22.39 Data Alignment in RSENTnSRXD Register**

| RSENTnCC.SMF | RSENTnSRXD.SMGC | RSENTnSRXD.IDD[19:16] | RSENTnSRXD.IDD[15:12] | RSENTnSRXD.IDD[11:8] | RSENTnSRXD.IDD[7:4] | RSENTnSRXD.IDD[3:0] |
|---|---|---|---|---|---|---|
| $00_B$ | Undefined | Undefined | Undefined | Undefined | Undefined | Status and communication nibble |
| $01_B$ | Undefined | Undefined | Undefined | Message ID[3:0] | DATA[7:4] | DATA[3:0] |
| $10_B$ | 0 | Message ID[7:4] | Message ID[3:0] | DATA[11:8] | DATA[7:4] | DATA[3:0] |
| $10_B$ | 1 | Message ID[3:0] | DATA[15:12] | DATA[11:8] | DATA[7:4] | DATA[3:0] |

### (6) Slow Channel Reception Flow

In **Figure 22.18**, the recommended reception flow for the slow channel receive buffer is shown. When the slow channel receive data is required, this process should be executed as part of the fast channel reception flow.

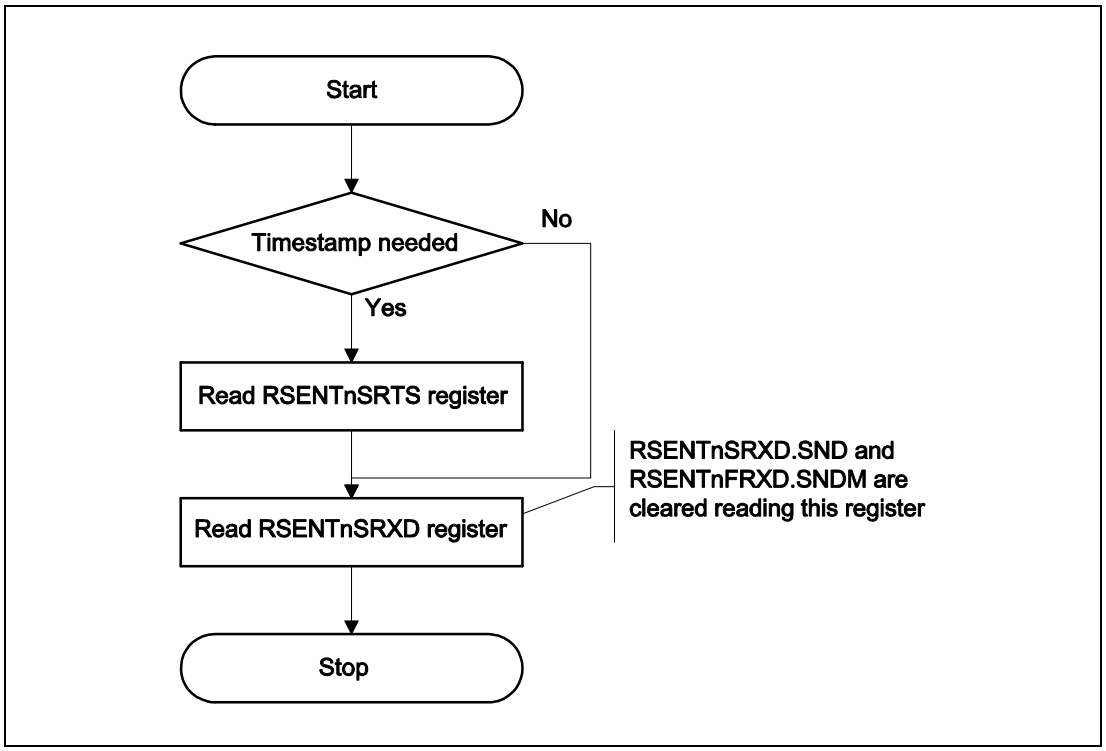



Figure 22.18 Slow Channel Reception Flow

In any case, the CPU should keep the order in reading the slow channel receive buffer registers as shown in the flow. The RSENTnSRXD.SND bit should be accessed as last.

### (7) DMA Flow

In case of DMA usage, the start address for the DMA usage and the number of transfers define which part of the receive buffer will be transferred. The RSENTnFRXD register should be the last register to be accessed using a 32 bit access method.

In case of SAE communication with pause pulse and fixed message length, the flow must be extended by checking of the ratio of the calibration pulse to message length. This variation check must be performed by the CPU. In case the variation check fails, the CPU must discard the received message.

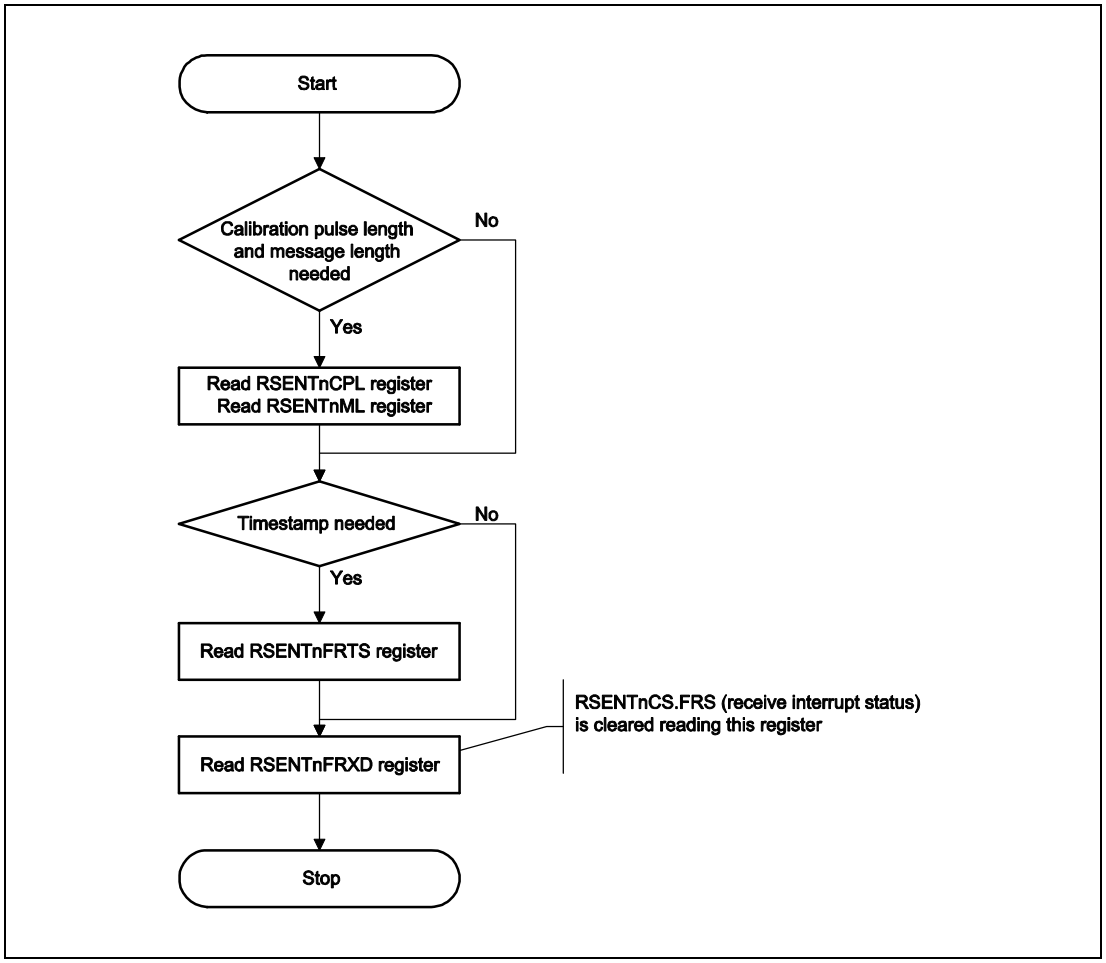



**Figure 22.19 DMA Reception Flow**

In the software processing, when the transferred data set, the CPU should check the status of the transferred RSENTnFRXD.SNDM bit. If this bit is set to 1, then the user needs to read the slow channel receive buffer if needed.

### (8) Error Flagging

Message lost errors (shown in RSENTnCS.SMS, RSENTnCS.FMS) are flagged when a new message was decoded and all message diagnostics passed.

The SPC communication error (shown in RSENTnCS.NRS) is flagged when the CPU writes to RSENTnSPCT.TLL before/during response reception.

The update timings for fast channel reception errors (RSENTnCS.CVS, RSENTnCS.CLS, RSENTnCS.FNS, RSENTnCS.FES and RSENTnCS.FCS) and slow channel reception errors (RSENTnCS.SCS and RSENTnCS.SES) depends on the configuration of RSENTnCC.SPCE, RSENTnCC.FCM, RSENTnCC.PPC and RSENTnCC.PPTC. The update timings for each configuration are shown in **Table 22.40** and **Table 22.41**.

In case a nibble encoding error or calibration pulse length error is detected, message reception is terminated immediately. No further error flagging for this message is done. Message decoding starts again after a calibration pulse without calibration length error (RSENTnCS.CLS) is detected.

**Table 22.40 Error flag set timing when CC.SPCE = 0**

| **RSENTnCC.SPCE** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
|---|---|---|---|---|---|---|---|---|
| **RSENTnCC.FCM** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |
| **RSENTnCC.PPC** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **RSENTnCC.PPTC** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** |
| RSENTnCS.FCS | EC | X | EC | IM | IM | X | IM | IM |
| RSENTnCS.FES | EC | X | EC | IM | IM | X | IM | IM |
| RSENTnCS.FNS | EC | X | EC | — | — | X | — | — |
| RSENTnCS.SCS | IM | X | IM | IM | IM | X | IM | IM |
| RSENTnCS.SES | IM | X | IM | IM | IM | X | IM | IM |
| RSENTnCS.CLS | IM | X | IM | IM | IM | X | IM | IM |
| RSENTnCS.CVS | EC | X | EC | — | EC | X | EC | — |

EC: End of calibration pulse
IM: Immediately when detected
—: Not set
X: Invalid configuration

**Table 22.41 Error flag set timing when CC.SPCE = 1**

| **RSENTnCC.SPCE** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
|---|---|---|---|---|---|---|---|---|
| **RSENTnCC.FCM** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |
| **RSENTnCC.PPC** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **RSENTnCC.PPTC** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** |
| RSENTnCS.FCS | IM | X | IM | IM | IM | X | IM | IM |
| RSENTnCS.FES | IM | X | IM | IM | IM | X | IM | IM |
| RSENTnCS.FNS | — | X | — | — | — | X | — | — |
| RSENTnCS.SCS | IM | X | IM | IM | IM | X | IM | IM |
| RSENTnCS.SES | IM | X | IM | IM | IM | X | IM | IM |
| RSENTnCS.CLS | IM | X | IM | IM | IM | X | IM | IM |
| RSENTnCS.CVS | — | X | — | — | — | X | — | — |

EC: End of calibration pulse
IM: Immediately when detected
—: Not set
X: Invalid configuration

**NOTE**

In case the sensor stops communication, no buffer update or status update for last message takes place. The SW should take care of this by timeout checks.

When a transition to OPERATION IDLE is configured in RSENTnMDC.OMC and an error is detected in the message in which the mode transition was requested, the error is not flagged and the message is aborted.

In case of a fast channel encoding error or a calibration pulse length error the OPERATION IDLE mode is entered immediately.

In case of a fast channel nibble count error, fast channel CRC error, or fast channel calibration pulse variation error the OPERATION IDLE state is entered at the end of the next STATUS/COM nibble.

RERSENTnCS.FNS is only set after a valid calibration pulse was detected and all following nibbles have a valid length (≥12 ticks and ≤27 ticks) or no nibble was received between two valid calibration pulses.

RSENTnCS.FES is only set if the nibble with an encoding error occurred in the communication and status nibble, CRC nibble or in one of the expected data nibbles.

If SPC is enabled (RSENTnCC.SPCE = 1), RSENTnCS.CLS is set if a calibration pulse was expected but the pulse length does not meet the calibration pulse range. If SPC is disabled (RSENTnCC.SPCE = 0), RSENTnCS.CLS is set only after a valid calibration pulse has been received and a calibration pulse was expected but the pulse length does not meet the calibration pulse range.

During re-synchronisation additional error flags might be set which is not affecting the reception of the following frame

### 22.4.4 SPC Function

The RSENT module supports an extension of the J2716 specification known as SPC. The user can enable or disable the SPC extension by setting the RSENTnCC.SPCE bit.

In SPC mode the RX line will be driven low to initiate RSENT message transmission. In the RH850/P1x-C this can be realized by controlling an external transistor by the RSENTnSPCO pin.

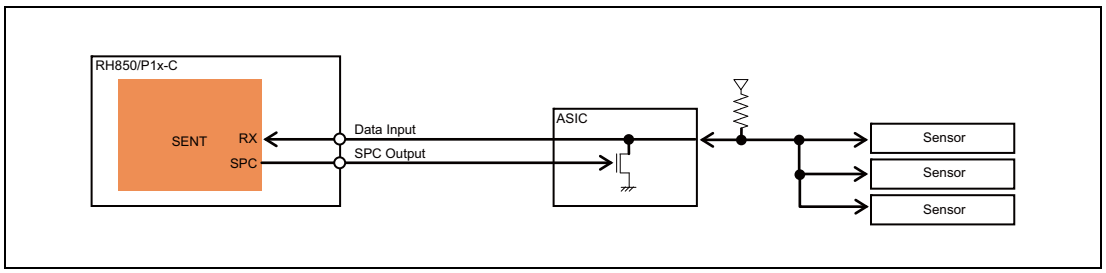



**Figure 22.20 Sample Circuit of External Transistor**

The user can configure the polarity of the Port RSENTnSPCO.

The text below describes the behavior of the RSENTnSPCO port with the default settings of RSENTnCC.SOPC. When the default value of RSENTnCC.SOPC is changed, the polarity is inverted.

RX line will be held low for a configured length of tick time specified in the RSENTnSPCT.TLL bits.The Tick time is configured with the RSENTnBRP.TTI bits and the RSENTnBRP.TTF bits which are equal to the transmission tick time. For details, see **Section 22.4.2.2(2), RX and SPC Tick Settings**.

In a single sensor system, this function can be used to trigger data transmission from the sensor. Further data can be sent to the sensor by varying the trigger pulse length. In a multi sensor system, this function can be used to address a dedicated sensor and request a data transmission.

Once RSENT SPC initialization is complete, transmission can be triggered by writing the trigger pulse width to the RSENTnSPCT.TLL register. When a transmission is triggered, the trigger pulse with the configured length is sent. Then a frame reception is expected. After frame reception was done, a new trigger pulse can be sent.

Writing to RSENTnSPCT.TLL requests a SPC trigger transmission. After writing to RSENTnSPCT.TLL, the CPU should read RSENTnCS.NRS to check whether the previous request was completed or not.

In case RSENTnCS.NRS is set, no SPC trigger is sent and any potentially ongoing reception at this time is aborted. The CPU should clear RSENTnCS.NRS by writing 1 to RSENTnCSC.NRC. The CPU can write again to RSENTnSPCT.TLL to request a SPC trigger transmission.

In case RSENTnCS.NRS is not set, the CPU should set a reception timeout counter in software. If a reception occurs before the timeout counter elapses, the user should process the received slow and fast channel data as shown in the fast channel reception flow (**Figure 22.15**) and slow channel reception flow (**Figure 22.18**).

When the timeout counter elapses without any successful reception, the addressed sensor seems not to send any valid response. The CPU should analyze the RSENTnCS register to analyze the reason for no successful reception. A new request can be made considering that when RSENTnCS.NRS gets set no SPC trigger is sent.

According to the SPC protocol, a frame is received when the falling edge of the end pulse has been detected. It is the user responsibility to not make a SPC trigger request during the end pulse as this might not be recognized by the sensor.

Purpose of the timeout function is to define a timeout window for response reception in software.

**Figure 22.21** shows a transmission flow with a timeout function implemented in software. The timeout function is optional and can be omitted if not needed.

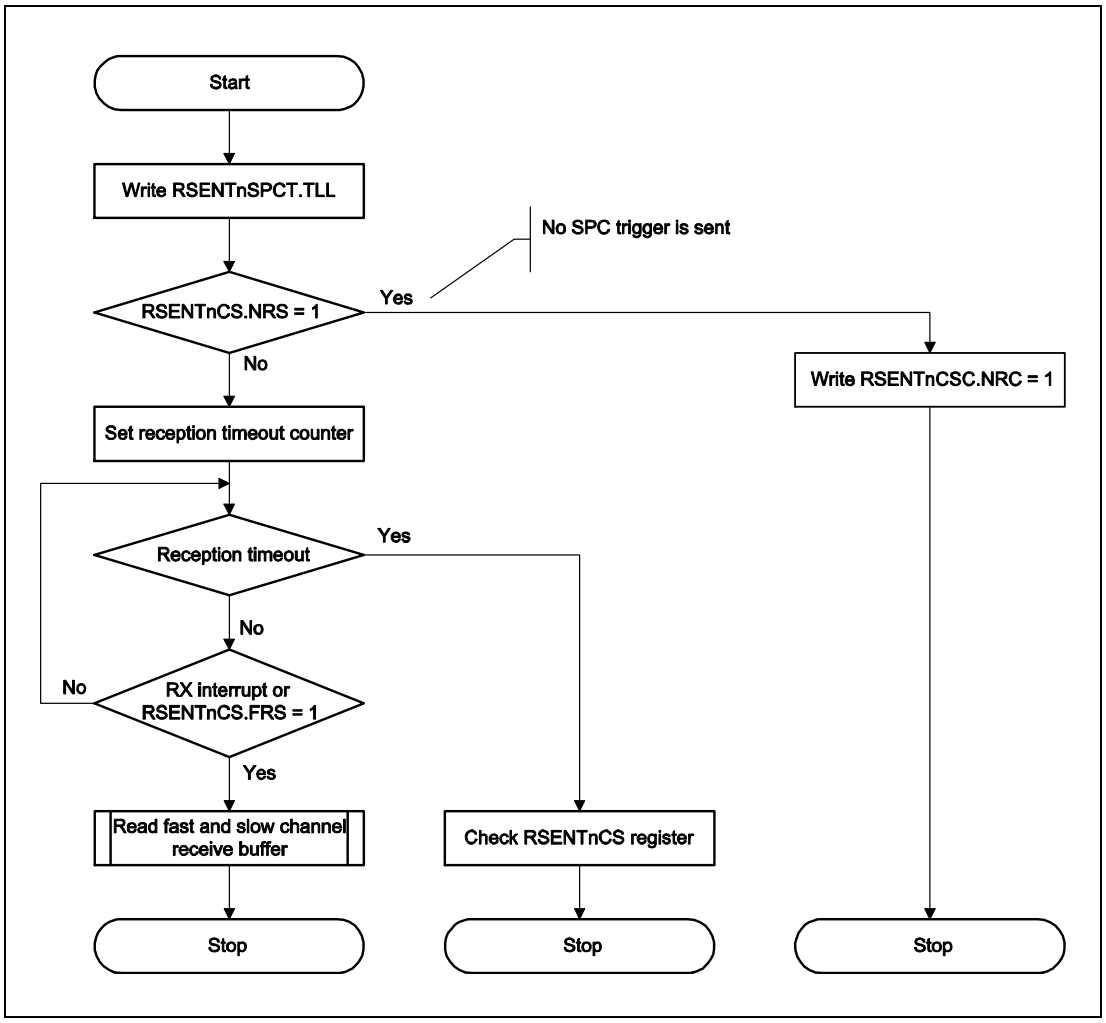



**Figure 22.21 Transmission Flow**

### 22.4.5 Interrupts and Checks

The RSENT module provides two interrupt lines.

The successful fast channel receive interrupt notifies the CPU that the fast channel receive buffer was updated and is holding a set of valid received data. Also, the reception status bit is set (RSENTnCS.FRS).

The status interrupt notifies the CPU that at least one of the error flags or message lost flags in the RSENTnCS register is set.

Whether a status flag in the RSENTnCS register is contributing to the generation of an interrupt event or not can be set individually.

The execution of the CRC checks can be disabled for the slow channel and fast channel individually. In case a check is disabled, the CRC of the received message is not checked and the related error flag is never set.

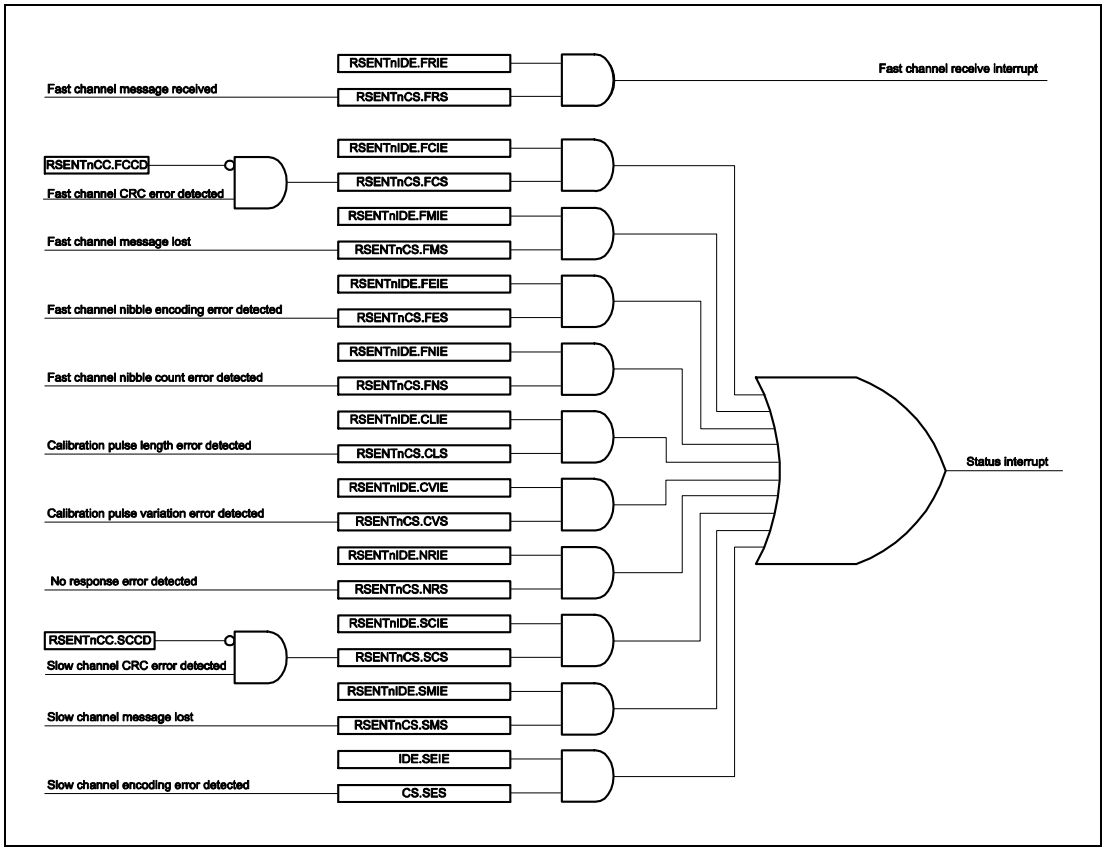



**Figure 22.22 Interrupt Structure**

**Table 22.42** gives an overview about the relationship between set status flags and the buffer update.

**Table 22.42 Status Flag Influence to Receive Buffer Behavior**

| RSENTnCS | Fast Channel Receive Buffer | Slow Channel Receive Buffer |
|---|---|---|
| FRS | Updated | Updated if last slow channel nibble received and RSENTnCS.SES = 0 and RSENTnCS.SCS = 0 |
| FCS | Not updated | Not impacted. Fast channel CRC does not cover the communication and status nibble |
| FMS | Background buffer overwritten | Not impacted |
| FES | Not updated | Receive process aborted. Search for new start condition |
| FNS | Not updated | Receive process aborted. Search for new start condition |
| CLS | Not updated | Receive process aborted. Search for new start condition |
| CVS | Not updated | Receive process aborted. Search for new start condition |
| NRS | Not updated | Receive process aborted. Search for new start condition |
| SCS | Not impacted | Not updated |
| SMS | Not impacted | Message lost |
| SES | Not impacted | Receive process aborted. Search for new start condition |

### 22.4.6 Limited Reset and Module stand-by

The RSENTn can be reset by limited reset from SYSCTRL. The Limited Reset is functionally equivalent to a hardware reset. Software must ensure that RSENTn is halted (RESET Mode or CONFIGURATION Mode). (See **Section 22.4.1, Modes of Operation**)

The RSENT clock can be disabled by the SYSCTRL module stand-by function. Software must ensure that RSENT is RESET mode or CONFIGURATION mode if module stand-by enable.

## 22.5 Difference among P1M-C, P1H-C and P1H-CE

Table 22.43 Different specification of RSENT

| Device | P1M-C (QFP, BGA-292) | P1M-C (BGA-156) | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|---|---|
| RSENT channel | 6 | 4 | 8 | 4 | 8 | 10 |

# Section 23 Window Watchdog Timer A (WDTA)

## 23.1 Features

The purpose of the Window Watchdog Timer A (WDTA) is to detect deadlock of CPU operation. WDTA is implemented as 1 or 2 macros in each device. One monitors so that PE1 won't be going out of control. Also, other monitors PE2. This timer generates an error signal if the counter is not cleared by CPU within a certain counter value. If the timer is cleared, MCU can keep normal operation. Additionally to detect failure mode, when the CPU feeds data for clearing the watchdog timer counter in incorrect timing due to unexpected CPU operation, extra feature are necessary such as a window trigger, VAC(Varying Activation Code) etc.

### 23.1.1 Number of Units and Channels

Table 23.1 Number of WDTA channel

| | Device | | | |
|---|---|---|---|---|
| **Macro** | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
| WDTA0 | 1 | 1 | 1 | 1 |
| WDTA1 | 0 | 1 | 1 | 1 |

### 23.1.2 Register Base Address

WDTAn base addresses are listed in the following table. WDTAn register addresses are given as offsets from the base addresses in general.

Table 23.2 Register Base Address

| Base Address Name | Base Address |
|---|---|
| <WDTA0_base> | FFED $0000_H$ |
| <WDTA1_base> | FFED $1000_H$ |

### 23.1.3 Clock Supply

Table 23.3 Clock Supply

| Unit Name | Specification | Description |
|---|---|---|
| WDTA0/ WDTA1 | High speed system clock : CLK_HSB | APB Bus clock |
| WDTA0/ WDTA1 | WDTA counter clock : WDTACLKI | WDT count clock*1<br>1/1 or 1/32 of CLK_IOSC*2 |

Note 1. Selectable by using the corresponding option byte

Note 2. See **Section 12, Clock Controller**

### 23.1.4 Interrupt Request

WDTAn interrupt requests are listed in the following table.

**Table 23.4 Interrupt Requests**

| Unit Name | Interrupt Name | Interrupt Number | DMA Number | Description |
|---|---|---|---|---|
| WDTA0 | INTWDTA0 | 8 | — | WDTA0 75% interrupt for PE1 |
| WDTA0 | | | | |
| WDTA1 | INTWDTA1 | 8 | — | WDTA1 75% interrupt for PE2 |
| WDTA1 | | | | |

**Table 23.5 Internal Error Signal**

| Unit Name | Interrupt for Unit | Description | Connected to |
|---|---|---|---|
| WDTA0 | WDTA0TERR*1 | WDTA0 error signal | ECM |
| WDTA1 | WDTA1TERR*1 | WDTA1 error signal | ECM |

Note 1. For details about error input to ECM, refer to **Section 29, Error Control Module (ECM)**

### 23.1.5 External Input / Output Pins

WDTAn has no external pins

## 23.2 Overview

### 23.2.1 Functional Overview

The Window Watchdog Timer A generates the ECM error signal if the 16-bit counter overflows or if any other error condition is fulfilled. The counter is cleared and restarted every time a WDTA trigger occurs while the window is open.
At 75% of the maximum counter value, the WDTA can generate an interrupt request INTWDTA0 and INTWDTA1. After reset release, the start-up options specify the start mode and the WDTA settings. The settings can be modified by writing the Watchdog Timer mode register WDTAnMD.

The Window Watchdog Timer A has the following functions:

- Operation mode after reset selectable by using start-up options
- Fixed activation code and variable activation code (VAC) selectable
- Two start modes available:
  - Default start mode (automatic start)
  - Software trigger start mode
- Generation of an error signal to ECM on error detection
- Interrupt request generation at 75% of the counter overflow value
- Window function

### 23.2.2 Block Diagram

**Figure 23.1** shows the main components of the Window Watchdog Timer A

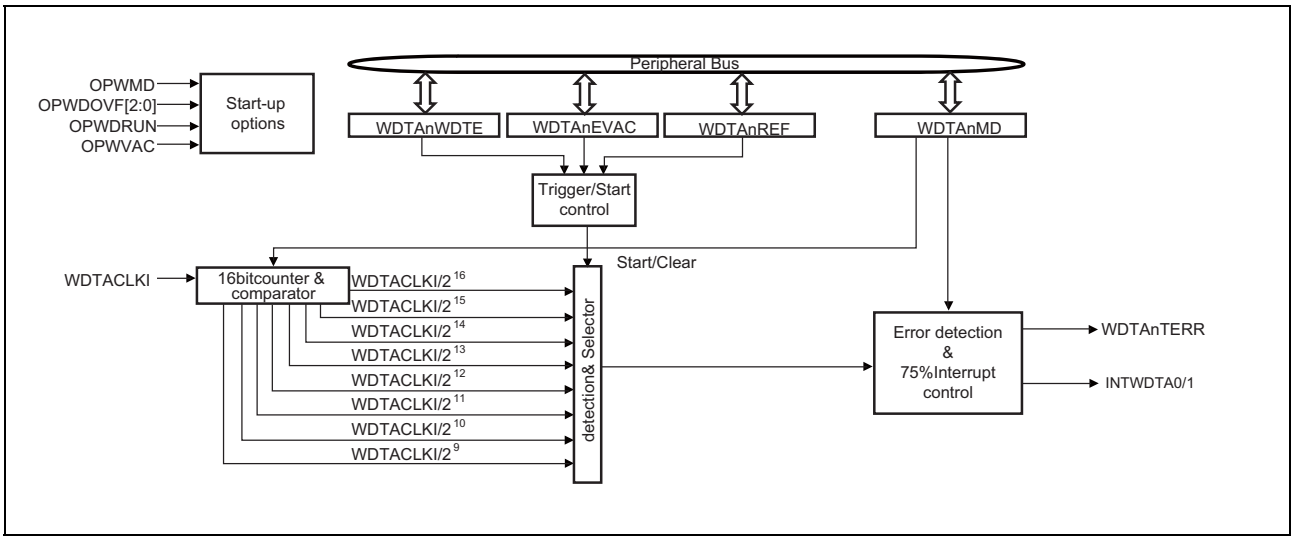



**Figure 23.1 Block diagram of the Window Watchdog Timer A**

# 23.3 Registers

## 23.3.1 List of Registers

WDTAn registers are listed in the following table.

**Table 23.6 Register list**

| Address | Register name | Description | Access Size[bit] | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | Other |
| * | * | WDTA0 | 8 | * | PBG2.PG2-WDT0 | — |
| * | * | WDTA1 | 8 | * | PBG2.PG2-WDT1 | — |

*: Regarding address, register name, and value after reset of WDTAn, see the following table

**Table 23.7 Register Reset Condition**

| Register Name | Reset condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset1 | System Reset2 | Application Reset1 | Limited Reset |
| All registers | √ | √ | √ | √ | — |

**Table 23.8 List of WDTn Registers**

| Module | Register | Symbol | Address |
|---|---|---|---|
| WDTAn | WDTA enable register | WDTAnWDTE | <WDTAn_base> + $0000_H$ |
| WDTAn | WDTA VAC enable register | WDTAnEVAC | <WDTAn_base> + $0004_H$ |
| WDTAn | WDTA reference value register | WDTAnREF | <WDTAn_base> + $0008_H$ |
| WDTAn | WDTA mode register | WDTAnMD | <WDTAn_base> + $000C_H$ |

## 23.3.2 WDTAnWDTE — WDTA enable register

This register is the WDTA start control and trigger register if the VAC function is not used (start-up option OPWDVAC = 0).

**NOTE**

See the **Table 23.17, WDTA start-up options**.

### WDTA trigger

Writing $AC_H$ to this register restarts the counter.The behavior of this register depends on activation of the VAC function, see **Table 23.11, WDTAnWDTE behavior**.

**Access:** This register can be read/written in 8-bit units.

**Address:** <WDTAn_base> + $0000_H$

**Value after reset:** The value after reset depends on the start-up options OPWDEN, OPWDRUN and OPWDVAC. See **Table 23.10, WDTAnRUN value after reset**.
This register is initialized by a reset of any type.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WDTAnRUN | — | — | — | — | — | — | — |
| Value after reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 23.9 WDTAnWDTE register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | WDTAnRUN | Enables/disables the WDTAn:<br>0: WDTAn disabled<br>1: WDTAn enabled<br>Since the WDTA can not be stopped once it was started, this bit can only be cleared by a reset. |
| 6 to 0 | Reserved | When read, the value after reset is read. |

This bit is only valid if VAC is disabled (OPWDVAC = 0). In this case, the initial value of bit WDTAnRUN depends on other start-up options as listed below.

**Table 23.10 WDTAnRUN value after reset**

| Start-up options | | |
|---|---|---|
| **OPWDVAC** | **OPWDRUN** | **Value after reset of WDTAnRUN** |
| 0 | 1 | 1 |
| Other than above | | 0 |

The behavior of WDTAnWDTE during read/write accesses depends on activation of the VAC mode, as shown in **Table 23.11** below.

**Table 23.11 WDTAnWDTE behavior**

| | WDTAnWDTE | | |
|---|---|---|---|
| **OPWDVAC** | **Read** | **Write** | **Remark** |
| 0 | $AC_H$ | WDTA trigger $AC_H$*1 | VAC disabled, WDTAnWDTE enabled |
| 1 | $2C_H$ | ignored | VAC enabled, WDTAnWDTE disabled |

Note 1. Any other write value will lead to an error detection.

### 23.3.3 WDTAnMD — WDTA mode register

This register specifies the overflow interval time, the 75% interrupt enable/disable, and the window-open period.

It can be updated only once after reset release and before the first trigger. The updated value is effective after the next WDTA trigger.

Updating this register after the WDTA has been started leads to error detection, but the read value of this register can be written without generating an error.

**Access:** This register can be read/written in 8-bit units.

**Address:** <WDTAn_base> + $000C_H$

**Value after reset:** The value after reset depends on the start-up options OPWDOVF[2:0].
This register is initialized by a reset of any type.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | WDTAnOVF[2:0] | | | WDTAnWIE | — | WDTAnWS[1:0] | |
| Value after reset | 0 | — | — | — | 0 | 1 | 1 | 1 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 23.12 WDTAnMD register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 to 4 | WDTAnOVF[2:0] | Selects the overflow interval time (μs) |

| WDTAn OVF2 | WDTAn OVF1 | WDTAn OVF0 | Overflow interval time |
|---|---|---|---|
| 0 | 0 | 0 | $2^9$ / WDTACLKI |
| 0 | 0 | 1 | $2^{10}$ / WDTACLKI |
| 0 | 1 | 0 | $2^{11}$ / WDTACLKI |
| 0 | 1 | 1 | $2^{12}$ / WDTACLKI |
| 1 | 0 | 0 | $2^{13}$ / WDTACLKI |
| 1 | 0 | 1 | $2^{14}$ / WDTACLKI |
| 1 | 1 | 0 | $2^{15}$ / WDTACLKI |
| 1 | 1 | 1 | $2^{16}$ / WDTACLKI |

The reset values of WDTAnOVF[2:0] depend on start-up option OPWDOVF[2:0].

| | | |
|---|---|---|
| 3 | WDTAnWIE | Enables/disables the 75% interrupt request INTWDTA0/1:<br>0: INTWDTA0/1 disabled(value after reset)<br>1: INTWDTA0/1 enabled |
| 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | WDTAnWS[1:0] | Selects the window-open period: |

| WDTAn WS1 | WDTAn WS0 | Window-open period |
|---|---|---|
| 0 | 0 | 25% |
| 0 | 1 | 50% |
| 1 | 0 | 75% |
| 1 | 1 | 100% (Default) |

### 23.3.4 WDTAnEVAC — WDTA enable VAC register

This register is the start control and trigger register if the VAC function is used (start-up option OPWDVAC = 1).

#### WDTA trigger

Writing the correct activation code to this register restarts the counter.

The behavior of this register depends on activation of the VAC function. See **Table 23.15, WDTAnEVAC behavior**.

**Access:** This register can be read/written in 8-bit units.

**Address:** <WDTAn_base> + $0004_H$

**Value after reset:** The value after reset depends on the start-up options OPWDEN,OPWRUN and OPWDVAC. See **Table 23.14, WDTAnEVAC value after reset**
This register is initialized by a reset of any type.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WDTAnEVAC7 | — | — | — | — | — | — | — |
| Value after reset | — | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R | R |

**Table 23.13 WDTAnEVAC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | WDTAnEVAC7 | Enables/disables the WDTAn:<br>0: WDTAn disabled<br>1: WDTAn enabled<br>Since the WDTA cannot be stopped once it was started, this bit can only be cleared by a reset. Thus even if bit 7 of the activation code is 0, the WDTA will not stop. |
| 6 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

This bit is only valid if VAC is enabled (OPWDVAC = 1). In this case, the value after reset of the WDTAnEVAC7 bit depends on other start-up options as listed below.

**Table 23.14 WDTAnEVAC value after reset**

| Start-up options | | |
|---|---|---|
| **OPWDRUN** | **Start mode** | **Value after reset of WDTAnEVAC7** |
| 0 | Software trigger | 0 |
| 1 | Default | 1 |
| Ignored | Software trigger | 0 |
| Ignored | Default | 1 |

The behavior of WDTAnEVAC during read/write accesses depends on activation of the VAC mode, as shown in **Table 23.15** below.

Table 23.15 WDTAnEVAC behavior

| OPWDVAC | WDTAnEVAC | | Remark |
|---|---|---|---|
| | Read | Write | |
| 0 | $2C_H$ | Ignored | VAC disabled |
| 1 | Last written VAC | ExpectWDTE*1 | VAC enabled |

Note 1. Any other write value will lead to an error detection.

## 23.3.5 WDTAnREF — WDTA reference value register

This register contains the reference value for calculating the activation code of the VAC function. It is automatically updated after every trigger operation.

If VAC is disabled (OPWDVAC = 0), reading this register returns $00_H$.

**Access:** This register can be read/written in 8-bit units.

**Address:** <WDTAn_base> + $0008_H$

**Value after reset:** $00_H$
This register is initialized by a reset of any type.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WDTAnREF[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

Table 23.16 WDTAnREF register contents

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | WDTAnREF[7:0] | Reference value for activation code calculation. |

## 23.4 Operation

### 23.4.1 WDTA Start-up Options

The start-up options determine the start-up configuration of the WDTA after reset release. They are described in **Table 23.17, WDTA start-up options**.

**Table 23.17 WDTA start-up options**

| Start-up option | Function | Description | Connected to |
|---|---|---|---|
| OPWD0MD | WDTA slow/fast mode | Switches between slow and fast mode:<br>0: WDTA slow mode (WDTACLKI = 1/32 of CLK_IOSC)<br>1: WDTA fast mode (WDTACLKI = 1/1 of CLK_IOSC) | • WDTA0: Flash option*1<br>• WDTA1: Flash option*1 |
| OPWDOVF[2:0] | Count clock setting | Specifies the reset value of the count clock control bits WDTAnMD.WDTAnOVF[2:0]. | • WDTA0/WDTA1: Flash option*1 |
| OPWDRUN | Start mode setting | Specifies the start mode:<br>0: Software trigger start mode<br>1: Default start mode | • WDTA0: Flash option*1<br>• WDTA1: Flash option*1 |
| OPWDVAC | Variable Activation Code (VAC) selection | Specifies the trigger register for the generation of counter re-start triggers to keep the counter from overflowing.<br>0: WDTAnWDTE (fixed)<br>1: WDTAnEVAC (variable)<br><br>When WDTAnWDTE is selected, the value to be written to the register (activation code) is fixed ($AC_H$). When WDTAnEVAC is selected, the activation code to be written to the register is variable. | • WDTA0: Flash option*1<br>• WDTA1: Flash option*1 |

Note 1. Refer **Section 32.12**

**NOTE**

The Window Watchdog Timer A generates an error signal for ECM if an error occurred. If the ECM is configured to generate a reset as a result on this error signal, it is indicated in a reset status register:

## 23.4.2 WDTA after reset release

### 23.4.2.1 Start modes

The WDTA provides two modes for the counter start after reset release:

- Software trigger start mode
  The counter value remains $0000_H$ after reset release.
  The counter is started with the first WDTA trigger.
- Default start mode
  The counter starts automatically after reset release.

### 23.4.2.2 Start mode selection

The start mode can be selected by the start-up options.

The start mode selection is listed in Table Start mode selection.

**Table 23.18 Start mode selection**

| Start-up options | | |
|---|---|---|
| **OPWDRUN** | **Reset type** | **Start mode** |
| 0 | Ignored | Software trigger |
| 1 | | Default |

### 23.4.2.3 WDTA settings after reset release

The WDTA settings are as follows between reset release and the first trigger:

**Table 23.19 WDTA settings after reset release**

| Function | Setting | Remark |
|---|---|---|
| Start mode | Specified by start-up options | |
| Count clock | | |
| 75% interrupt mode | Disabled | |
| Window-open period | 100% | If default start mode is specified, the first trigger is valid any time before the counter overflows. |

#### Change WDTA settings

After the first trigger, the WDTA continues according to the settings of the Watchdog Timer mode register WDTAnMD.

To change the WDTA settings, WDTAnMD must be written before the first trigger. Changing the value of WDTAnMD after the first trigger leads to an error.

If WDTAnMD is not changed before the first trigger, the WDTA mode is specified by the value after reset of WDTAnMD.

The new or value after reset of WDTAnMD applies after the first trigger.

### 23.4.2.4 Default start mode timing

The default start mode timing and the changes to the WDTA settings are illustrated in **Figure 23.2**.

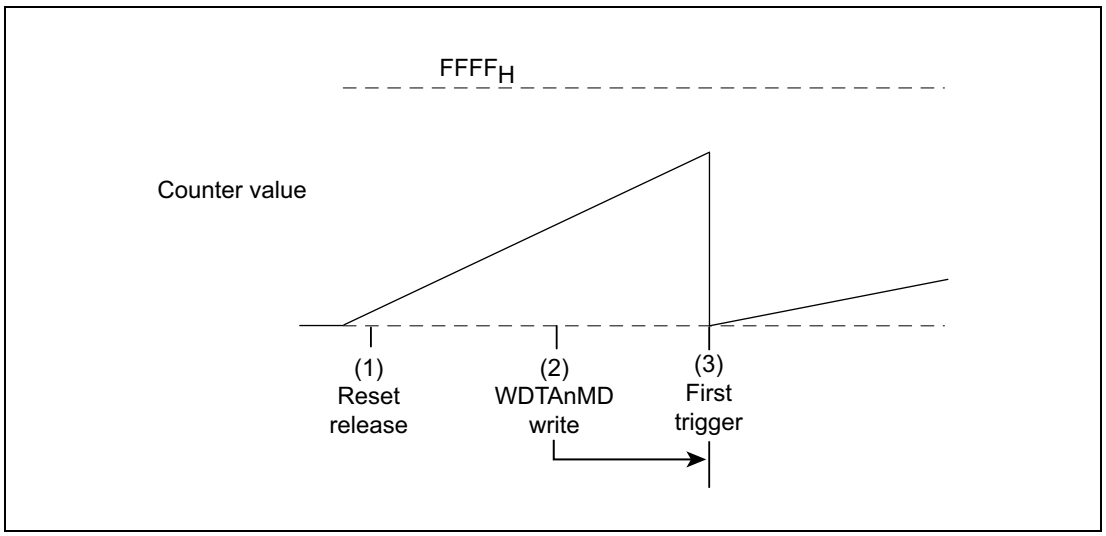



**Figure 23.2 Timing diagram of WDTA start in default start mode**

The timing diagram above shows the following:

(1) After reset release, the counter starts immediately.
The count clock is specified by the start-up options, for example:
– Count clock after reset release = WDTACLKI / $2^{13}$
(OPWDOVF[2:0] = $100_B$)

(2) WDTAnMD is written with WDTAnMD.WDTAnOVF[2:0] = $111_B$ before the first trigger.
However, the settings are not applied immediately.

(3) The first trigger must occur before the counter overflows.
After the first trigger, the settings specified in WDTAnMD are applied, for example a new count clock:
– Count clock after first trigger = WDTACLKI / $2^{16}$ (WDTAnMD.WDTAnOVF[2:0] = $111_B$

### 23.4.2.5 Software trigger start mode timing

The software trigger start mode timing and the changes to the WDTA settings are illustrated in **Figure 23.3**.

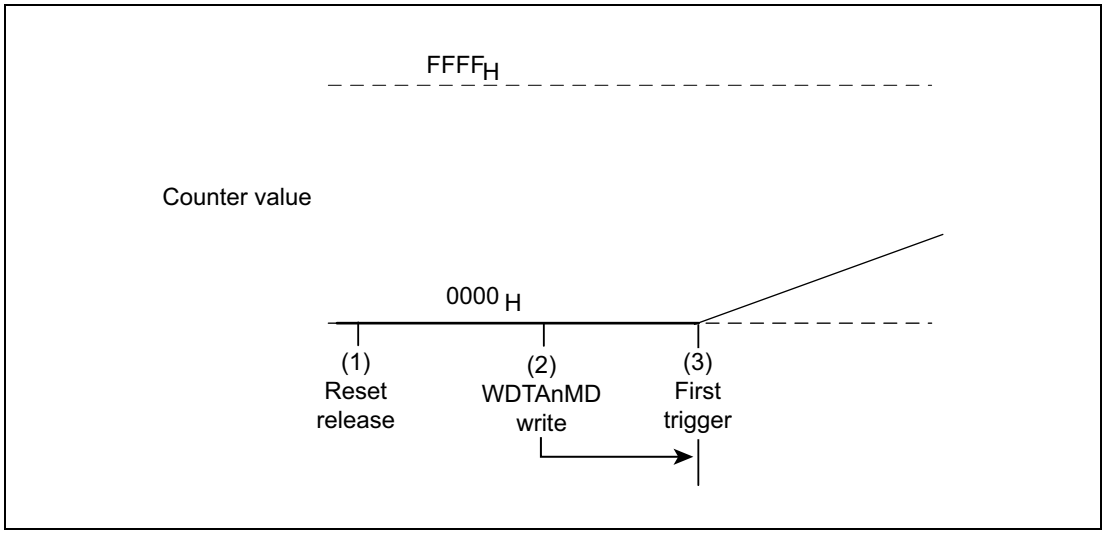



**Figure 23.3 Timing diagram of WDTA start in software trigger start mode**

The timing diagram above shows the following:

(1) After reset release, the counter remains $0000_H$ until the first trigger.
The count clock is specified by the start-up options, but it does not have any effect.

(2) WDTAnMD is written before the first trigger. However, the settings are not applied immediately.

(3) The counter starts at the first trigger.
The count clock and other settings specified in WDTAnMD are applied.

### 23.4.3 WDTA trigger

The WDTA trigger has the following functions:

- Counter start trigger in software trigger start mode
- Counter restart trigger to avoid counter overflow

The trigger register to be used differs depending on whether the activation code is fixed or variable. The type of activation code and the associated trigger register are specified by using the start-up option OPWDVAC.

**Table 23.20 Trigger register and activation code**

| Type of activation code | Trigger register | Activation code |
|---|---|---|
| Fixed | WDTAnWDTE | $AC_H$ |
| Variable | WDTAnEVAC | For details, refer to Calculation variable activation codes |

#### 23.4.3.1 Calculating variable activation codes

The variable activation code (ExpectWDTE) is calculated using a reference value in register WDTAnREF. The reference value in WDTAnREF is updated each time the trigger register WDTAnEVAC is written.

- Use the expression below to calculate the variable activation code (ExpectWDTE):
  ExpectWDTE = $AC_H$ – WDTAnREF (old)
- Use the expression below to calculate how the WDTAnREF value is updated:
  WDTAnREF (new) = rotate left 1 bit (ExpectWDTE)

The table below lists the variable activation codes according to the number of triggers.

**Table 23.21 Expected variable activation code development**

| No*1 | WDTAnREF (old) | | ExpectWDTE ($AC_H$ — WDTAnREF) | | WDTAnREF (new) | |
|---|---|---|---|---|---|---|
| 0 | 0000 0000 | $00_H$ | 1010 1100 | $AC_H$ | 0101 1001 | $59_H$ |
| 1 | 0101 1001 | $59_H$ | 0101 0011 | $53_H$ | 1010 0110 | $A6_H$ |
| 2 | 1010 0110 | $A6_H$ | 0000 0110 | $06_H$ | 0000 1100 | $0C_H$ |
| … | … | … | … | … | … | … |

Note 1. Number of triggers after reset

**NOTE**

Bit 7 of the WDTAnEVAC register (WDTAnEVAC7) cannot be cleared to 0 after the WDTA has been started. Thus even if bit 7 of the activation code is 0, the WDTA will not stop.

### 23.4.4 Error detection

The conditions for error detection are:

- Overflow interval time is exceeded (counter overflow)
- Wrong activation code is written to the trigger register
- Writing to the trigger register while the window is closed.
- Illegal update of Watchdog Timer mode register WDTAnMD:
  - Writing a new value to WDTAnMD after the first trigger leads to an error detection.
  - Writing the same value to WDTAnMD after the first trigger does not lead to an error detection.
- When an error is detected, an error signal for ECM (WDTAnTERR) is generated.

**Figure 23.4** shows the error signal to ECM generation when the counter overflows and default start mode is selected.

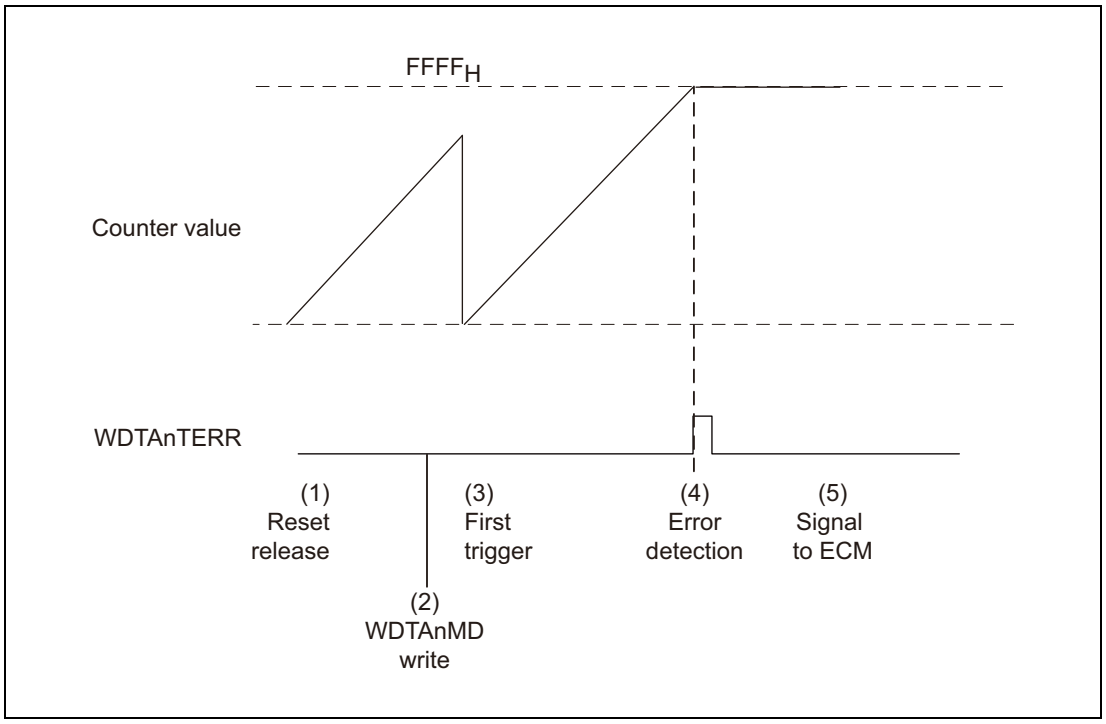



**Figure 23.4 Timing diagram of WDTA error signal to ECM module**

The timing diagram above shows the following:

(1) After reset release, the counter starts (default start mode is selected).

(2) WDTAnMD is written before the first trigger. However, the settings are not applied immediately.

(3) The counter is cleared at the first trigger and the new WDTA settings are applied.

(4) When the counter overflows, an error is detected. The WDTAnTERR is generated. The counter value remains until a reaction from ECM occurs.

### 23.4.5 75% interrupt request signals

When the counter reaches 75% of the maximum counter value, the interrupt request INTWDTAn is generated.

By use of WDTAnMD.WDTAnWIE this function can be enabled or disabled afterwards.

**Figure 23.5** shows the 75% interrupt request generation under following conditions:

- Default start mode selected
- Count clock changes after first trigger

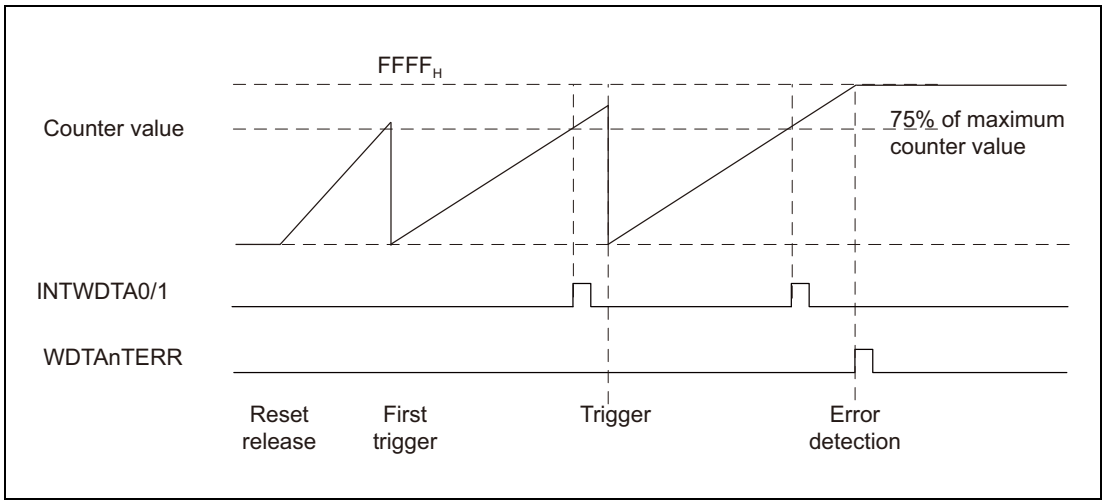



**Figure 23.5 Timing diagram of WDTA 75% interrupt request signals**

### 23.4.6 Window function

When the window-open period is set to less than 100%, an error is detected if the trigger occurs while the window is closed.

The definition of the window-open period differs before and after the first trigger:

- After reset release, the window-open period is 100%.
- After the first trigger, the window-open period is specified by bits WDTAnMD.WDTAnWS[1:0].

**Figure 23.6** shows WDTA operation with a window-open period of 25% and with default start mode selected.

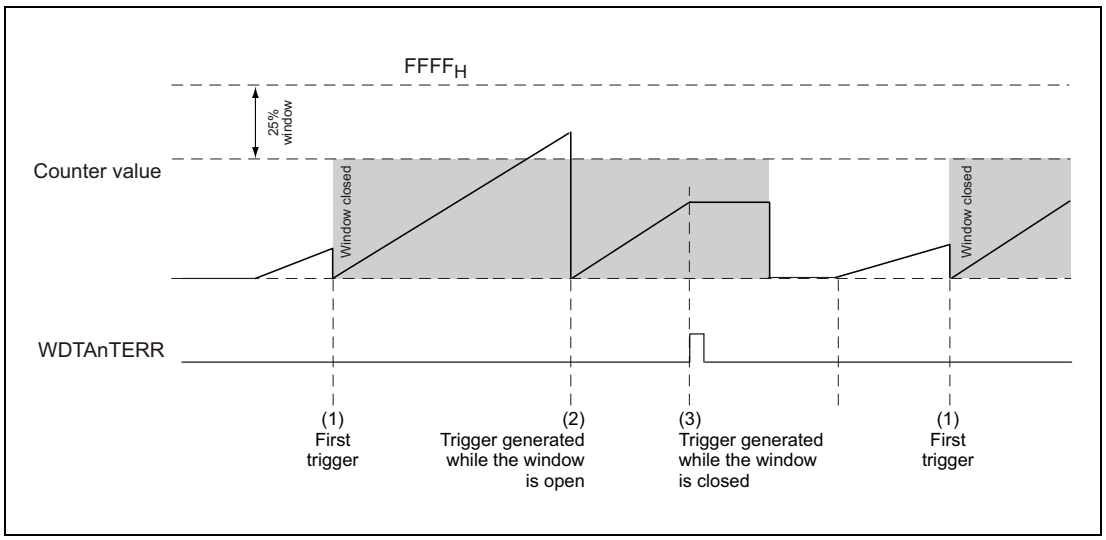



**Figure 23.6 Timing diagram of WDTA window function**

The timing diagram above shows the following:

(1) The window-open period is fixed to 100% for the first trigger.

(2) A trigger generated while the window is open does not lead to an error.

(3) A trigger generated while the window is closed generates an error signal to ECM module.

## 23.5 Cautions

1. Register access to WDTA0 is allowed from PE1 only. Do not access to WDTA0 from other masters.
   Register access to WDTA1 is allowed from PE2 only. Do not access to WDTA1 from other masters.
   Initial setting of each PBUS guard allows access from PE1/ PE2 to each WDTs. At first after start up, change the setting of PBUS guard for WDT0 to allow access from PE1 only. Also, change the setting of PBUS guard for WDT1 to allow access from PE2 only.
2. When WDT is default (automatic) start mode, WDT counter setting as default start mode begins to count up at the same time with a PE start up even if other PE doesn't start fetch. Please be careful in order to clear a counter of each WDT before reaching default threshold. When WDT counter can't be cleared before reaching the threshold, please use WDT by a Software start mode.

## 23.6 Difference among P1M-C, P1H-C and P1H-CE

There are differences in channels, interrupts, and external pins. For details, see **Table 23.1**, and **Table 23.4**.

# Section 24 System Timer (STM)

## 24.1 Features

The system timer (STM) is intended for typical operating system tasks like scheduling and synchronization of SW processes and supervision of timing budgets in single and multi-core environments. It will also be used for generation of wait loops or synchronization of SW processes with HW events. A set of 4 hardware counter channels allows running typical OS services like task scheduling and time budget monitoring on individual time bases. This reduces efforts to emulate additional timer channels in SW normally derived from only one common hardware timer. Multiple compare registers per counter channel can be used to start cyclic tasks with a defined temporal offset. A dedicated wide 64-bit counter channel enables the generation of a unique timestamp covering long period operation periods without overflow (e.g. complete automotive driving cycle) and thus eliminates any additional overflow SW handling.

### 24.1.1 Number of Units and Channels

Table 24.1 Number of STM channel

| Macro | Device | | | | Description |
|---|---|---|---|---|---|
| | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE | |
| STM0 | 1 | 1 | 1 | 1 | STM0 connect to NTU clock from FlexRay. Interrupt 8 of STM0 is non-maskable interrupt for PE1. |
| STM1 | 0 | 1 | 1 | 1 | STM1 don't connect to NTU clock from FlexRay. Interrupt 8 of STM1 is non-maskable interrupt for PE2. |

### 24.1.2 Register Base Address

STMn base addresses are listed in the following table. STMn register addresses are given as offsets from the base addresses in general.

Table 24.2 Register Base Address

| Base Address Name | Base Address |
|---|---|
| <STM0_base> | FFDD 8000H |
| <STM1_base> | FFDD 9000H |

### 24.1.3 Clock Supply

Table 24.3 Clock Supply

| Unit Name | Specification | Description |
|---|---|---|
| STM0 | High speed system clock: CLK_HSB | Peripheral clock (PCLK) and Counter clock source |
| | FlexRay NTU clock | Counter clock source |
| STM1 | High speed system clock: CLK_HSB | Peripheral clock (PCLK) and Counter clock source |

### 24.1.4 Interrupt Request

STMn interrupt requests are listed in the following table.

Table 24.4 STMn interrupt requests

| Unit Name | Interrupt Name | Interrupt Number | DMA Number | Description |
|---|---|---|---|---|
| STM0 | INTSTM00 | 41 | — | — |
| STM0 | INTSTM01 | 42 | — | — |
| STM0 | INTSTM02 | 12 | — | — |
| STM0 | INTSTM03 | 13 | — | — |
| STM0 | INTSTM04 | 14 | — | — |
| STM0 | INTSTM05 | 15 | — | — |
| STM0 | INTSTM06 | 135 | — | — |
| STM0 | INTSTM07 | 136 | — | — |
| STM0 | FEINT for PE1 | FEINT | — | Non-maskable interrupt for PE1 |
| STM1 | INTSTM10 | 43 | — | — |
| STM1 | INTSTM11 | 44 | — | — |
| STM1 | INTSTM12 | 45 | — | — |
| STM1 | INTSTM13 | 46 | — | — |
| STM1 | INTSTM14 | 137 | — | — |
| STM1 | INTSTM15 | 138 | — | — |
| STM1 | INTSTM16 | 139 | — | — |
| STM1 | INTSTM17 | 140 | — | — |
| STM1 | FEINT for PE2 | FEINT | — | Non-maskable interrupt for PE2 |

### 24.1.5 External Input / Output Pins

STMn has no external pins

## 24.2 Overview

### 24.2.1 Functional overview

- 1x64-bit / 3x32-bit counter channel
- Atomic read/write access to all registers
- 4 compare channels per counter channel (bit width corresponding to counter channel)
- Free-run compare mode, counting up
- Clock sources: Peripheral Clock PCLK, FlexRay NTU clock
- 9 interrupts (8x maskable, 1x non maskable high priority)
- Configurable assignment of interrupts to compare and overflow events
- Common register to start / stop multiple counter channels synchronously by one single register access
- Common status register, reflecting all channels' compare match and overflow flags by one single register access
- Anytime write access to compare registers, anytime read access to counter registers
- Application reset 1 (SW reset) for individual channels can be masked. When masked, counter keeps running on reset occurrence and counter register will not be initialized.

### 24.2.2 Counter channels and compare registers

The System Timer is composed of one 64-bit channel and three 32-bit counter channels. Each counter is equipped with 4 compare registers with corresponding bit width.

The counter registers can be read at any time, both while counter is running or stopped. The compare registers can be written at any time.

The CPU architecture supports 64-bit write access on the 32-bit peripheral bus. However in case of individual 32-bit write accesses to 64-bit counter or compare registers of channel 0, the updated register value will be processed after write to the upper 32-bit register. The appropriate write sequence must therefore be specified by software (i.e., that the register must be accessed from the lower 32 bits, and then the upper 32 bits.)

In case of reading the 64-bit counter value while the counter is running, the current 64-bit counter value will be captured first and read access will be performed on the captured value then. This is to avoid that the upper or lower 32-bit value changes between the two consecutive 32-bit read accesses.

### 24.2.3 Block Diagram

The following block diagram shows the main components of the System Timer.

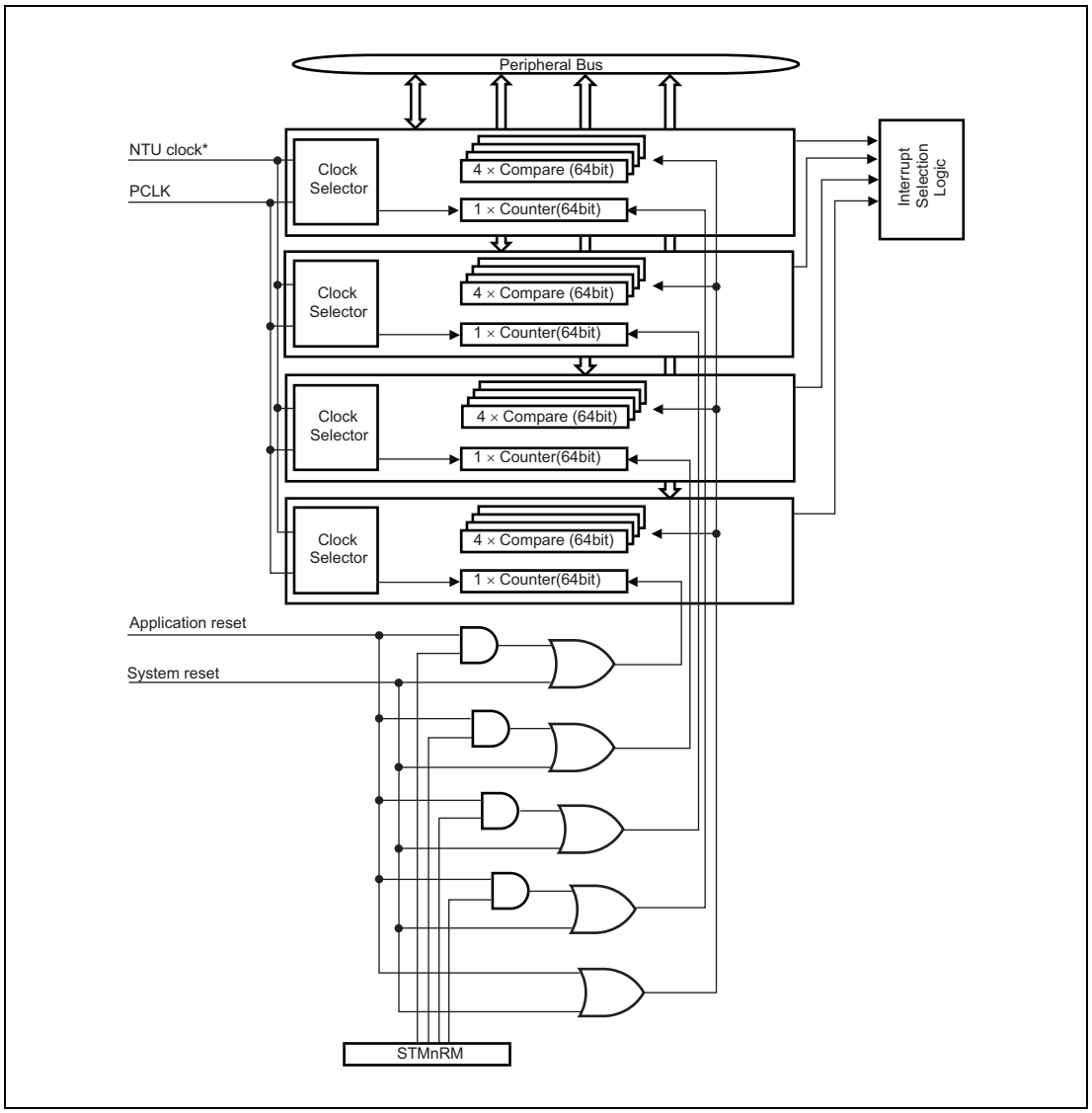



**Figure 24.1 STMn block diagram**

**NOTE**

STM1 does not support NTU clock function

## 24.3 Registers

This section contains a description of all registers of the System Timer.

### 24.3.1 List of Registers

The System Timer is controlled and operated by the following registers:

**Table 24.5 Register list**

| Address | Register name | Description | Access Size[bit] | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | Other |
| *1 | *1 | STM0 | 32 | *1 | PBG2.PG2-SSTM0 | — |
| *1 | *1 | STM1 | 32 | *1 | PBG2.PG2-SSTM1 | — |

Note 1. Regarding address, register name, and value after reset of STMn, see the following table

**Table 24.6 Register Reset Condition**

| Register Name | Reset condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset1 | System Reset2 | Application Reset1 | Limited Reset |
| STMnRM | √ | √ | √ | — | — |
| Other registers | √ | √ | √ | √ | — |

**Note:** “Other Registers” will only be reset if the corresponding channel is not masked in STMnRM register.

**Table 24.7 List of System Timer registers (1/2)**

| Channel | Register Name | Symbol | Address |
|---|---|---|---|
| CNT 0,1,2,3 Common Register | Timer Counter Source Clock Selection Register | STMnCKSEL*1 | <STMn_base> + $0000_H$ |
| | Timer Counter Start Register | STMnTS | <STMn_base> + $0010_H$ |
| | Timer Counter Stop Register | STMnTT | <STMn_base> + $0014_H$ |
| | Timer Counter Status Register | STMnCSTR | <STMn_base> + $0018_H$ |
| | Timer Status Register | STMnSTR | <STMn_base> + $0020_H$ |
| | Timer Status Clear Register | STMnSTC | <STMn_base> + $0024_H$ |
| | Timer Interrupt Selection Register | STMnIS | <STMn_base> + $0030_H$ |
| | Timer SW Reset Mask Register | STMnRM | <STMn_base> + $0034_H$ |
| CNT 0 | Timer Counter Register L | STMnCNT0L | <STMn_base> + $0040_H$ |
| | Timer Counter Register H | STMnCNT0H | <STMn_base> + $0044_H$ |
| | Timer Compare match Register AL | STMnCMP0AL | <STMn_base> + $0050_H$ |
| | Timer Compare match Register AH | STMnCMP0AH | <STMn_base> + $0054_H$ |
| | Timer Compare match Register BL | STMnCMP0BL | <STMn_base> + $0058_H$ |
| | Timer Compare match Register BH | STMnCMP0BH | <STMn_base> + $005C_H$ |
| | Timer Compare match Register CL | STMnCMP0CL | <STMn_base> + $0060_H$ |
| | Timer Compare match Register CH | STMnCMP0CH | <STMn_base> + $0064_H$ |
| | Timer Compare match Register DL | STMnCMP0DL | <STMn_base> + $0068_H$ |
| | Timer Compare match Register DH | STMnCMP0DH | <STMn_base> + $006C_H$ |

Table 24.7 List of System Timer registers (2/2)

| Channel | Register Name | Symbol | Address |
|---|---|---|---|
| CNT 1 | Timer Counter Register | STMnCNT1 | <STMn_base> + $0070_H$ |
| | Timer Compare match Register A | STMnCMP1A | <STMn_base> + $0080_H$ |
| | Timer Compare match Register B | STMnCMP1B | <STMn_base> + $0084_H$ |
| | Timer Compare match Register C | STMnCMP1C | <STMn_base> + $0088_H$ |
| | Timer Compare match Register D | STMnCMP1D | <STMn_base> + $008C_H$ |
| CNT 2 | Timer Counter Register | STMnCNT2 | <STMn_base> + $0090_H$ |
| | Timer Compare match Register A | STMnCMP2A | <STMn_base> + $00A0_H$ |
| | Timer Compare match Register B | STMnCMP2B | <STMn_base> + $00A4_H$ |
| | Timer Compare match Register C | STMnCMP2C | <STMn_base> + $00A8_H$ |
| | Timer Compare match Register D | STMnCMP2D | <STMn_base> + $00AC_H$ |
| CNT 3 | Timer Counter Register | STMnCNT3 | <STMn_base> + $00B0_H$ |
| | Timer Compare match Register A | STMnCMP3A | <STMn_base> + $00C0_H$ |
| | Timer Compare match Register B | STMnCMP3B | <STMn_base> + $00C4_H$ |
| | Timer Compare match Register C | STMnCMP3C | <STMn_base> + $00C8_H$ |
| | Timer Compare match Register D | STMnCMP3D | <STMn_base> + $00CC_H$ |

Note 1. STM1 does not have STM1CKSEL register.

## 24.3.2 STMnCKSEL — STMn timer counter source clock selection register

**Access:** STMnCKSEL register can be read/written in 32-/8-bit units.

**Address:** <STMn_base> + $0000_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by a reset of any type.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CKSEL 3 | CKSEL 2 | CKSEL 1 | CKSEL 0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

Table 24.8 STMnCKSEL register contents

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | CKSELm | This bit selects the input clock for counter channel m (m = 0 to 3).<br>0: PCLK (Peripheral clock)<br>This clock is dependent from the configured CPU and peripheral bus clock.<br>1: FlexRay NTU clock<br>This clock is derived from the FlexRay IP's network time unit. It can be used to synchronize SW processes (e.g. AUTOSAR job list) with the Flexray network. See the FlexRay chapter for a detailed description of the NTU clock generation. |

### 24.3.3 STMnTS — STMn timer counter start register

**Access:** STMnTS register can be written in 32-/8-bit units.

**Address:** <STMn_base>+ $0010_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TS3 | TS2 | TS1 | TS0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | W | W | W | W |

**Table 24.9 STMnTS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When writing, write the value after reset. |
| 3 to 0 | TSm | This bit starts the counter channel m.<br>0: No function<br>1: Starts the counter and sets STMnCSTR.CSTm = 1.<br>Setting this bit is ignored as long as STMnCSTR.CSTm = 1. |

**NOTES**

1. A reset from any source will initialize the bits unless the masking function has been selected in the STMn SW reset mask register (STMnRM) for channel m. If masking function is selected, the function related to register bits TSm is not affected by SW reset (application reset 1).
2. STMnTS is read always as $0000\ 0000_H$.

### 24.3.4 STMnTT — STMn timer counter stop register

**Access:** STMnTT register can be written in 32-/8-bit units.

**Address:** <STMn_base> + $0014_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TT3 | TT2 | TT1 | TT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | W | W | W | W |

**Table 24.10 STMnTT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When writing, write the value after reset. |
| 3 to 0 | TTm | This bit stops the counter channel m.<br>0: No function<br>1: Stops the counter and clears STMnCSTR.CSTm bit.<br>Setting this bit is ignored as long as STMnCSTR.CSTm = 0. |

**NOTES**

1. A reset from any source will initialize the bits unless the masking function has been selected in the STMn SW reset mask register (STMnRM) for channel m. If masking function is selected, the function related to register bits TTm is not affected by SW reset (application reset 1)
2. STMnTT is read always as $0000\ 0000_H$.

### 24.3.5 STMnCSTR — STMn timer counter status register

**Access:** STMnCSTR register can be read in 32-/8-bit units.

**Address:** <STMn_base> + $0018_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CST3 | CST2 | CST1 | CST0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 24.11 STMnCSTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. |
| 3 to 0 | CSTm | This bit indicates whether the counter is enabled or disabled.<br>0: Counter disabled<br>1: Counter enabled<br>This bit is set to 1 in response to STMnTS.TS being set to 1.<br>Setting STMnTT.TTn to 1 re-sets this bit to 0. |

**NOTE**

A reset from any source will initialize the bits unless the masking function has been selected in the STMn SW reset mask register (STMnRM) for channel m. If masking function is selected, the function related to register bits CSTm is not affected by SW reset (application reset 1).

### 24.3.6 STMnSTR — STMn timer status register

**Access:** STMnSTR register can be read in 32-/8-bit units.

**Address:** <STMn_base> + $0020_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | OVF3 | OVF2 | OVF1 | OVF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CM3D | CM3C | CM3B | CM3A | CM2D | CM2C | CM2B | CM2A | CM1D | CM1C | CM1B | CM1A | CM0D | CM0C | CM0B | CM0A |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 24.12 STMnSTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset is read. |
| 19 to 16 | OVFm | Overflow flag of timer channel m (m = 0 to 3)<br>0: No overflow has occurred<br>1: Overflow has occurred<br>Once set to 1 by overflow event, this bit remains 1 until cleared by SW write to overflow reset bit in STMn status clear register (STMnSTC.OVRm). |
| 15 to 0 | CMml | This bit indicates whether a compare match has occurred on compare register l (l = A, B, C, D) of timer channel m (m = 0 to 3)<br>0: No compare match has occurred<br>1: Compare match has occurred<br>Once set to 1 by compare match, this bit remains 1 until cleared by SW write to the compare match reset bit in STMn status clear register (STMnSTC.CMRml). |

**NOTE**

A reset from any source will initialize the bits unless the masking function has been selected in the STMn SW reset mask register (STMnRM) for channel m. If masking function is selected, the function related to register bits CSTm is not affected by SW reset (application reset 1).

### 24.3.7 STMnSTC — STMn timer status clear register

**Access:** STMnSTC register can be written in 32-/8-bit units.

**Address:** <STMn_base> + $0024_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by a reset of any type.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | OVR3 | OVR2 | OVR1 | OVR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMR3D | CMR3C | CMR3B | CMR3A | CMR2D | CMR2C | CMR2B | CMR2A | CMR1D | CMR1C | CMR1B | CMR1A | CMR0D | CMR0C | CMR0B | CMR0A |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 24.13 STMnSTC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When writing, write the value after reset. |
| 19 to 16 | OVRm | Reset overflow flag of timer channel m (m = 0 to 3)<br>0: no function<br>1: Reset overflow flag (STMnSTR.OVF = 0) |
| 15 to 0 | CMRml | Reset compare match flag of compare register l (l = A, B, C, D) of timer channel m (m = 0 to 3)<br>0: no function<br>1: Reset compare match event flag (STMnSTR.CMml = 0) |

**NOTE**

STMnSTC is read always as $0000\ 0000_H$.

In case an overflow or compare match interrupt occurs while the OVRm / CMRml flag is cleared by SW, the interrupt will have higher priority and set the flag.

### 24.3.8 STMnIS — STMn timer interrupt selection register

**Access:** STMnIS register can be read/written in 32-/8-bit units.

**Address:** <STMn_base> + $0030_H$

**Value after reset:** $0000\ 0000_H$ This register is initialized by a reset of any type.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STMINT7[1:0] | | STMINT6[1:0] | | STMINT5[1:0] | | STMINT4[1:0] | | STMINT3[1:0] | | STMINT2[1:0] | | STMINT1[1:0] | | STMINT0[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.14 STMnIS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When written, write the value after reset. When read, the value after reset is read. |
| 15 to 0 | STMINTi[1:0] | These bit groups select the system timer event to be signalled by STM interrupt i (i = 7 to 0).<br>Refer to Table **Table 24.15, Interrupt selection**. |

**Table 24.15 Interrupt selection**

| STMINTi[1:0] | STMINT0 | STMINT1 | STMINT2 | STMINT3 | STMINT4 | STMINT5 | STMINT6 | STMINT7 |
|---|---|---|---|---|---|---|---|---|
| 00 | CH0 A | CH0 B | CH0 C | CH0 D | CH1 A | CH1 B | CH1 C | CH1 D |
| 01 | CH2 A | CH2 B | CH3 A | CH3 B | CH2 A | CH2 B | CH3 A | CH3 B |
| 10 | Setting prohibited | Setting prohibited | Setting prohibited | OVF [CH0 v CH1 v CH2 v CH3] | Setting prohibited | Setting prohibited | Setting prohibited | OVF [CH0 v CH1 v CH2 v CH3] |
| 11 | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited |

**NOTE**

It is prohibited to map the same system timer event to multiple STMINT interrupts.

Interrupt INTn8 is not listed in this register overview as its assignment is not configurable.

Please refer also to **Section 24.4.1, Interrupt assignment**.

### 24.3.9 STMnRM — STMn timer SW reset mask register

**Access:** STMnRM register can be read/written in 32-/8-bit units.

**Address:** <STMn_base> + $0034_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | RM3 | RM2 | RM1 | RM0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 24.16 STMnRM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | RMm | This bit masks the SW reset (application reset 1) for counter channel (m = 0 to 3). When SW reset is masked for channel m it behaves as follows:<br>• The counter continues counting up operation. SW reset occurrence has no impact on the counter operation and the counter value.<br>• The count enable status will not be changed by SW reset; no stop trigger will be generated by occurrence of SW reset.<br>All other STM registers will be reset independent whether masking was selected or not. The input clock will be set to the default value PCLK.<br>0: Reset mask disabled for channel m<br>(Channel m responds to SW reset (application reset 1))<br>1: Reset mask enabled for channel m<br>(Channel m does not responds to SW reset (application reset 1)) |

**NOTE**

A reset from any source will initialize the bits unless the masking function has been selected in the STMn reset mask register (STMnRM) for channel m. If masking function is selected, the related register bits RMm will not be changed by SW reset (application reset 1).

## 24.3.10 STMnCNT0L — STMn timer counter register low (lower 32-bit)

**Access:** This register can be read/written in 32-/8-bit units.

**Address:** <STMn_base> + $0040_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNT0L[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNT0L[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.17 STMnCNT0L register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CNT0L[31:0] | Lower 32-bit of 64-bit counter value (channel 0) |

**NOTES**

1. A reset from any source will initialize the bits unless the masking function has been selected in the STMn SW reset mask register (STMnRM) for this channel. If masking function is selected, the register contents will not be changed by SW reset (application reset 1)
2. This register is part of a 64-bit register. The architecture supports 64-bit write access to the upper and lower register. However in case of individual 32-bit write accesses the total 64-bit register value will become effective after the higher 32-bit value has been written to. The appropriate write sequence must therefore be specified by software (i.e., that the register must be accessed from the lower 32 bits, and then the upper 32 bits.)
   Independent from access width, the user should use appropriate PBus-Guard setting to control the access to a 64-bit register. This is to avoid that during an access from one bus master (e.g. PE) another bus master (e.g. DMA) will access to the same register resulting in an invalid update of the register value.
3. Read access to counter registers is supported both while counter is running and stopped.
   In case of reading the 64-bit counter value while the counter is running, the current 64-bit counter value will be captured on read access to the lower 32-bit value. The captured upper 32-bit value can be read afterwards. This is to avoid that the upper or lower 32-bit value changes between the two consecutive 32-bit read accesses.
4. Write access is permitted only, while counter is stopped.

## 24.3.11 STMnCNT0H — STMn timer counter register high (upper 32-bit)

**Access:** This register can be read/written in 32-/8-bit units.

**Address:** <STMn_base> + $0044_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNT0H[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNT0H[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.18 STMnCNT0H register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CNT0H[31:0] | Upper 32-bit of 64-bit counter value (channel 0) |

**NOTES**

1. A reset from any source will initialize the bits unless the masking function has been selected in the STMn SW reset mask register (STMnRM) for this channel. If masking function is selected, the register contents will not be changed by SW reset (application reset 1)
2. This register is part of a 64-bit register. The architecture supports 64-bit write access to the upper and lower register. However in case of individual 32-bit write accesses the total 64-bit register value will become effective after the higher 32-bit value has been written to. The appropriate write sequence must therefore be specified by software (i.e., that the register must be accessed from the lower 32 bits, and then the upper 32 bits.)
   Independent from access width, the user should use appropriate PBus-Guard setting to control the access to a 64-bit register. This is to avoid that during an access from one bus master (e.g. PE) another bus master (e.g. DMA) will access to the same register resulting in an invalid update of the register value.
3. Read access to counter registers is supported both while counter is running and stopped.
   In case of reading the 64-bit counter value while the counter is running, the current 64-bit counter value will be captured on read access to the lower 32-bit value. The captured upper 32-bit value can be read afterwards. This is to avoid that the upper or lower 32-bit value changes between the two consecutive 32-bit read accesses.
4. Write access is permitted only, while counter is stopped.

## 24.3.12 STMnCMP0IL — STMn timer compare match register low (lower 32-bit)

**Access:** This register can be read/written in 32-/8-bit units.

**Address:** STMnCMP0AL:<STMn_base> + $0050_H$
STMnCMP0BL:<STMn_base> + $0058_H$
STMnCMP0CL:<STMn_base> + $0060_H$
STMnCMP0DL:<STMn_base> + $0068_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMP0IL[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMP0IL[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.19 STMnCMP0IL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CMP0IL[31:0] | Lower 32-bit of 64-bit compare value (channel 0, I = A, B, C, D) |

**NOTE**

This register is part of a 64-bit register. The architecture supports 64-bit write access to the upper and lower register. However in case of individual 32-bit write accesses the total 64-bit register value will become effective after the higher 32-bit value has been written to. The appropriate write sequence must therefore be specified by software (i.e., that the register must be accessed from the lower 32 bits, and then the upper 32 bits.)
Independent from access width, the user should use appropriate PBus-Guard setting to control the access to a 64-bit register. This is to avoid that during an access from one bus master (e.g. PE) another bus master (e.g. DMA) will access to the same register resulting in an invalid update of the register value.

## 24.3.13 STMnCMP0IH — STMn timer compare match register high (upper 32-bit)

**Access:** This register can be read/written in 32-/8-bit units.

**Address:** STMnCMP0AH:<STMn_base> + $0054_H$
STMnCMP0BH:<STMn_base> + $005C_H$
STMnCMP0CH:<STMn_base> + $0064_H$
STMnCMP0DH:<STMn_base> + $006C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMP0IH[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMP0IH[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.20 STMnCMP0IH register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CMP0IH[31:0] | Upper 32-bit of 64-bit compare value (channel 0, I = A, B, C, D) |

**NOTE**

This register is part of a 64-bit register. The architecture supports 64-bit write access to the upper and lower register. However in case of individual 32-bit write accesses the total 64-bit register value will become effective after the higher 32-bit value has been written to. The appropriate write sequence must therefore be specified by software (i.e., that the register must be accessed from the lower 32 bits, and then the upper 32 bits.)
Independent from access width, the user should use appropriate PBus-Guard setting to control the access to a 64-bit register. This is to avoid that during an access from one bus master (e.g. PE) another bus master (e.g. DMA) will access to the same register resulting in an invalid update of the register value.

### 24.3.14 STMnCNTm — STMn timer counter register

**Access:** This register can be read/written in 32-/8-bit units.

**Address:** STMnCNT1:<STMn_base> + $0070_H$
STMnCNT2:<STMn_base> + $0090_H$
STMnCNT3:<STMn_base> + $00B0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNTI[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNTI[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.21 STMnCNTm register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CNTm[31:0] | 32-bit counter value of channel m (m = 1 to 3) |

**NOTES**

1. A reset from any source will initialize the bits unless the masking function has been selected in the STMn SW reset mask register (STMnRM) for this channel. If masking function is selected, the register contents will not be changed by SW reset (application reset 1).
2. Read access to counter registers is supported both while counter is running and stopped.
3. Write access is permitted only, while counter is stopped.

## 24.3.15 STMnCMPml — STMn timer compare match register

**Access:** This register can be read/written in 32-/8-bit units.

**Address:** STMnCMP1A: <STMn_base> + $0080_H$, STMnCMP1B: <STMn_base> + $0084_H$,
STMnCMP1C: <STMn_base> + $0088_H$, STMnCMP1D: <STMn_base> + $008C_H$,
STMnCMP2A: <STMn_base> + $00A0_H$, STMnCMP2B: <STMn_base> + $00A4_H$,
STMnCMP2C: <STMn_base> + $00A8_H$, STMnCMP2D: <STMn_base> + $00AC_H$,
STMnCMP3A: <STMn_base> + $00C0_H$, STMnCMP3B: <STMn_base> + $00C4_H$,
STMnCMP3C: <STMn_base> + $00C8_H$, STMnCMP3D: <STMn_base> + $00CC_H$,

**Value after reset:** $0000\ 0000_H$ This register is initialized by a reset of any type.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPml[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPml[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 24.22 STMnCMPml register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CMPml[31:0] | 32-bit of compare value of channel m (m = 1 to 3), compare register l ( l = A, B, C, D) |

# 24.4 Operation

## 24.4.1 Interrupt assignment

The System Timer (STM) provides 9 individual interrupt signals, one out of these (INT8) as non-maskable with high priority (Note: non-maskable high-priority interrupt cannot be linked to channel 0 and OVF event). The assignment of interrupts to the compare registers and counter channel's overflag signal can be configured in the Interrupt Selection

Logic as described in **Figure 24.1, STMn block diagram**, **Table 24.23, Configurable Interrupt vector assignment**.The overflow signals of channel 0-3 will be ORed and mapped to one common, selectable interrupt vector. The ORed compare signals of channel 1 are fix assigned to one common non-maskable interrupt (INTn8). Additioonally they can be assigned to further non-maskable interrupts

For details on interrupt controller, see "Interrupt Functions".

**Table 24.23 Configurable Interrupt vector assignment**

| | | Selectable Interrupt Vector | | Fix Interrupt vector |
|---|---|---|---|---|
| **STM channel** | **Interrupt source** | **Alt0** | **Alt1** | **FEINT** |
| 0 (64-bit) | Compare match A | INTn0 | — | — |
| | Compare match B | INTn1 | — | — |
| | Compare match C | INTn2 | — | — |
| | Compare match D | INTn3 | — | — |
| 1 (32-bit) | Compare match A | INTn4 | — | INTn8 |
| | Compare match B | INTn5 | — | |
| | Compare match C | INTn6 | — | |
| | Compare match D | INTn7 | — | |
| 2 (32-bit) | Compare match A | INTn0 | INTn4 | — |
| | Compare match B | INTn1 | INTn5 | — |
| | Compare match C | — | — | — |
| | Compare match D | — | — | — |
| 3 (32-bit) | Compare match A | INTn2 | INTn6 | — |
| | Compare match B | INTn3 | INTn7 | — |
| | Compare match C | — | — | — |
| | Compare match D | — | — | — |
| 0 | ORed OVF flags channel 0 to 3 [OVF0 v OVF1 v OVF2 v OVF3] | INTn3 | INTn7 | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

Note 1. FE-level maskable interrupt (FEINT)

Note 2. INTn8 occurence can be masked as described in **Section 6.8.7, FEINT Source selection**

## 24.4.2 Cautions

### 24.4.2.1 Interrupt cautions

#### (1) Interrupt after reset

After the reset of the STM (PRESET or SW reset which not be masked), both counters and compare match registers are initialized to 0. So when you enable the counter in STM directly after the reset, the compare match interrupt request will be sent out and the flag of compare match will also be set.You must ignore the compare match requests and clear the compare match flags first. The above case could be avoided by setting the compare match registers with any value but 0 before you enable the counters after the reset of STM.

#### (2) Interrupt request when corresponding flag has been set

STM0STR is the status register in STM, the interrupt request according to the compare match event or overflow event within STM will be not send out if the corresponding flag in STM0STR has been already set. But if the interrupt request conflicts with the clear of the corresponding flag, the interrupt request will be send out.

### 24.4.2.2 Overlap of one-pulse interrupt

Generally the output interrupt requests of STM are one-pulse signals but in the case below the OVFI and INT8_STM will not be one-pulse signals. So you should take care of those cases in STM.

case1: OVFI overlapped for several cycles

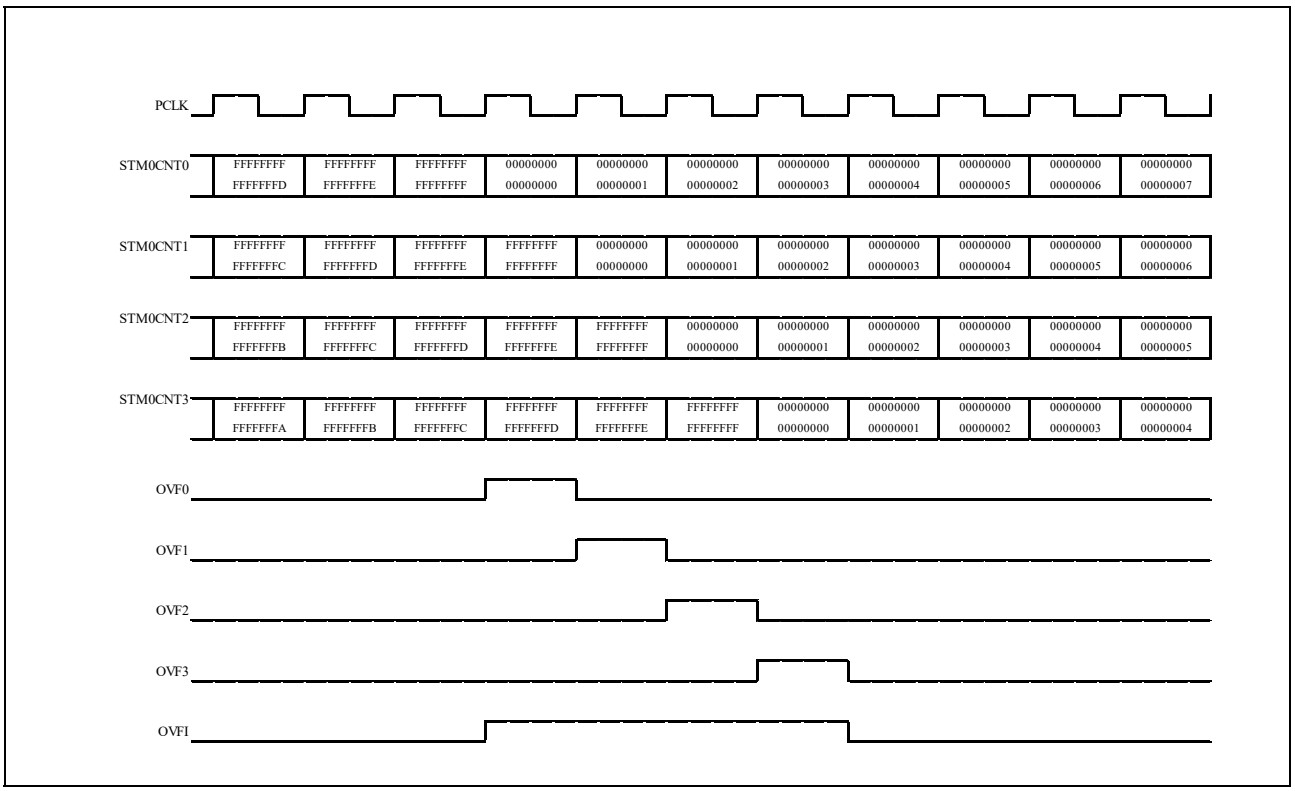



**Figure 24.2 OVFI overlapped for several cycles**

Case2: INT8_STM overlapped for several cycles

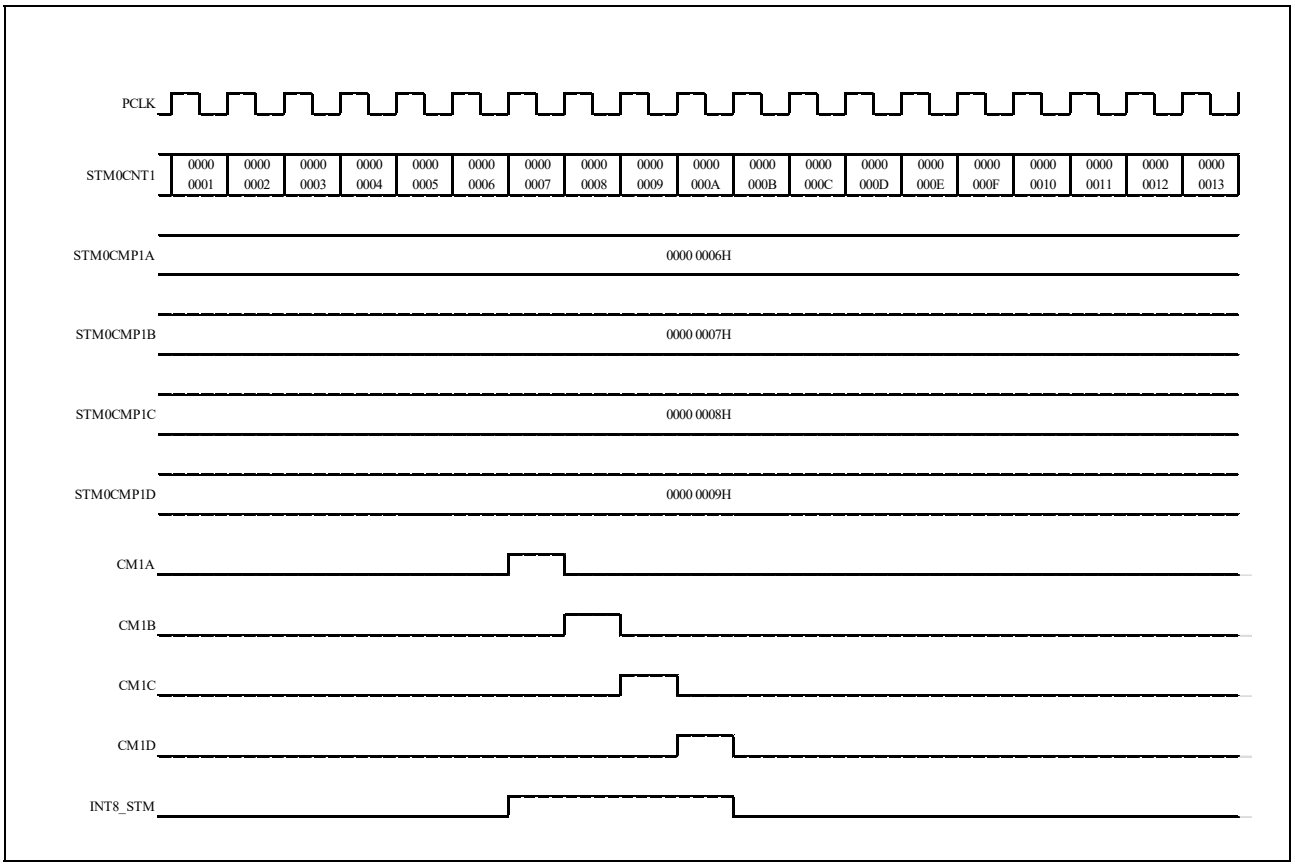



**Figure 24.3 INT8_STM overlapped for several cycles**

## 24.4.3 Attention in using

### 24.4.3.1 Confliction of compare match with writing of compare register

When compare match is conflicting with the writing of compare register, the compare match flag will be set with the consideration of not losing the compare event, and the compare match register will be written in too.

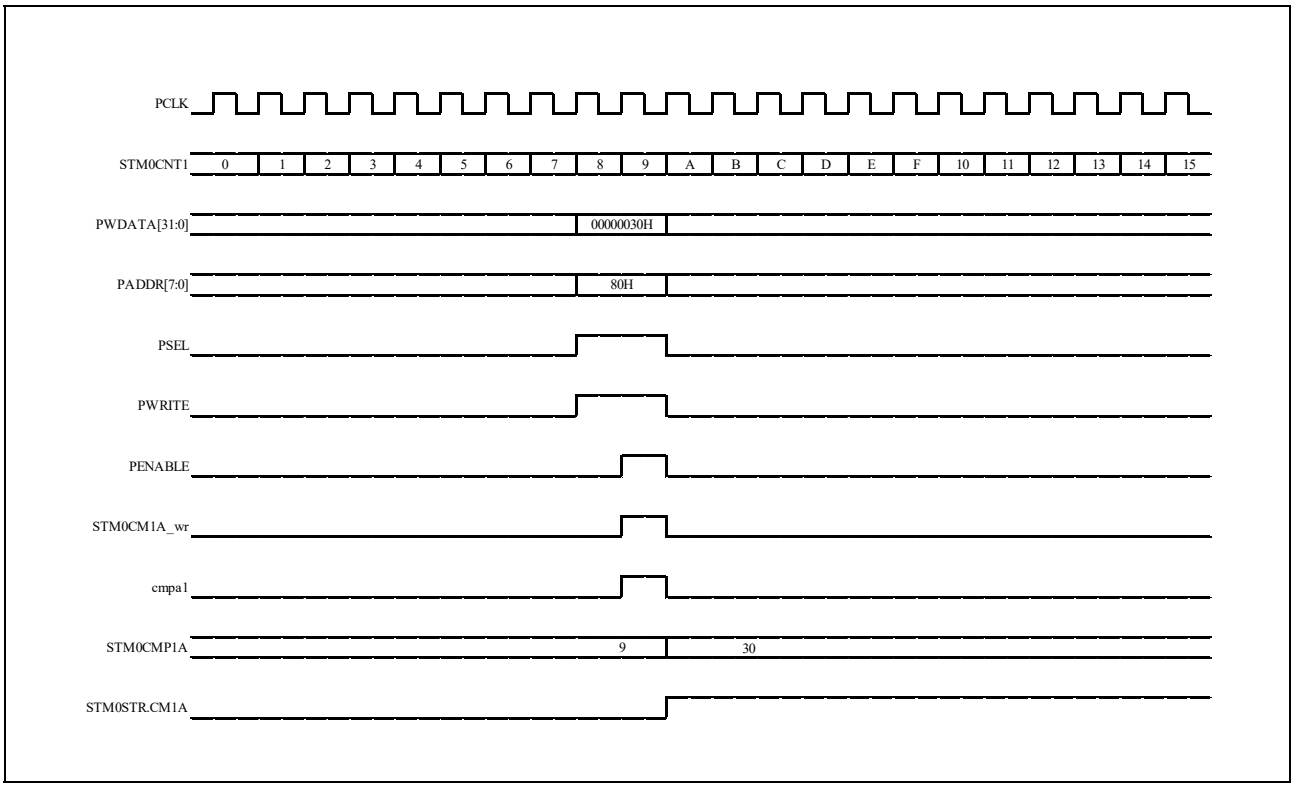



**Figure 24.4 Compare match event conflicting with write access (counting with PCLK)**

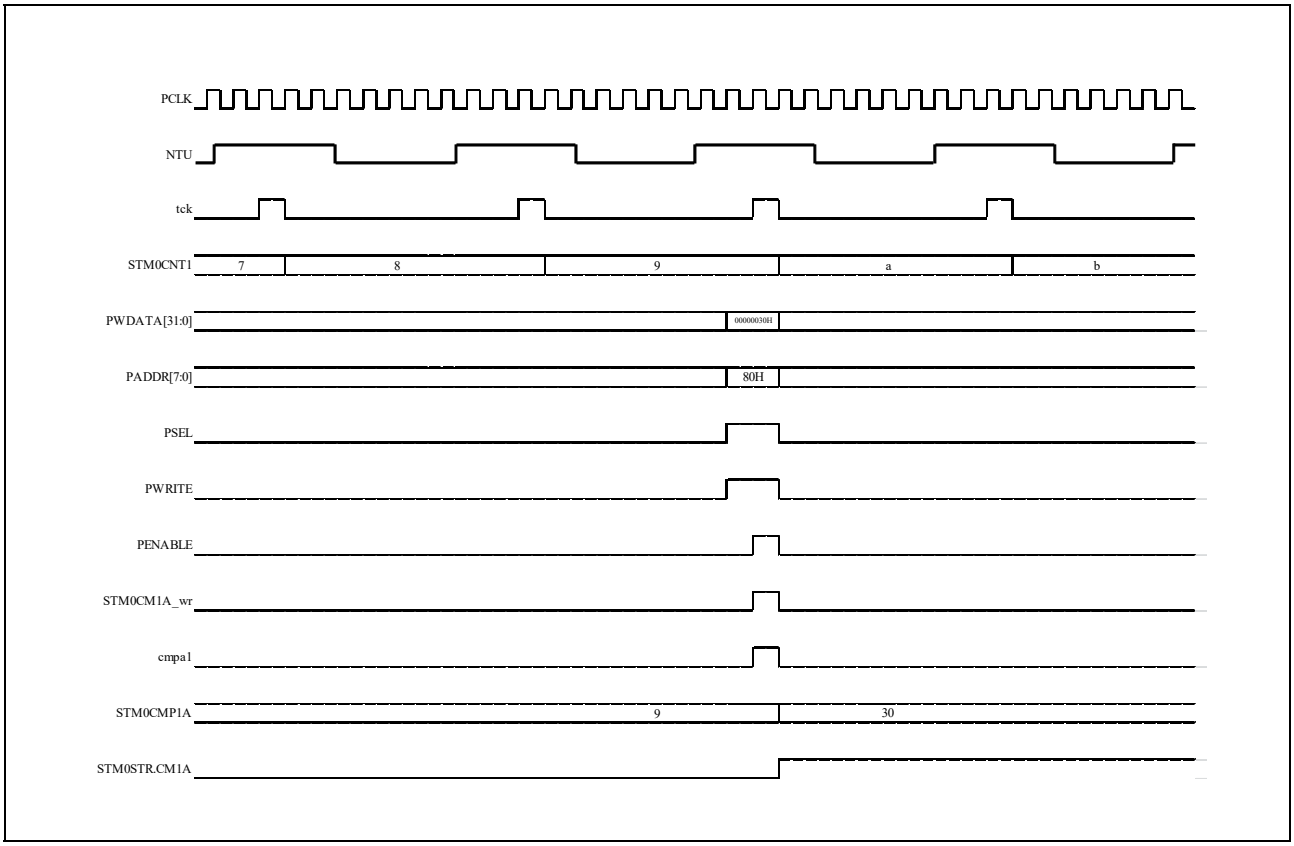



**Figure 24.5 Compare match event conflicting with write**

### 24.4.3.2 Confliction of compare match flag(overflow flag) set with its clear

When the compare match (overflow) flag's set conflict with its clear, compare match (overflow) flag will be set but the clear will be ignored whether the flag is 1 or 0 at that moment.You can get details from the below figure.

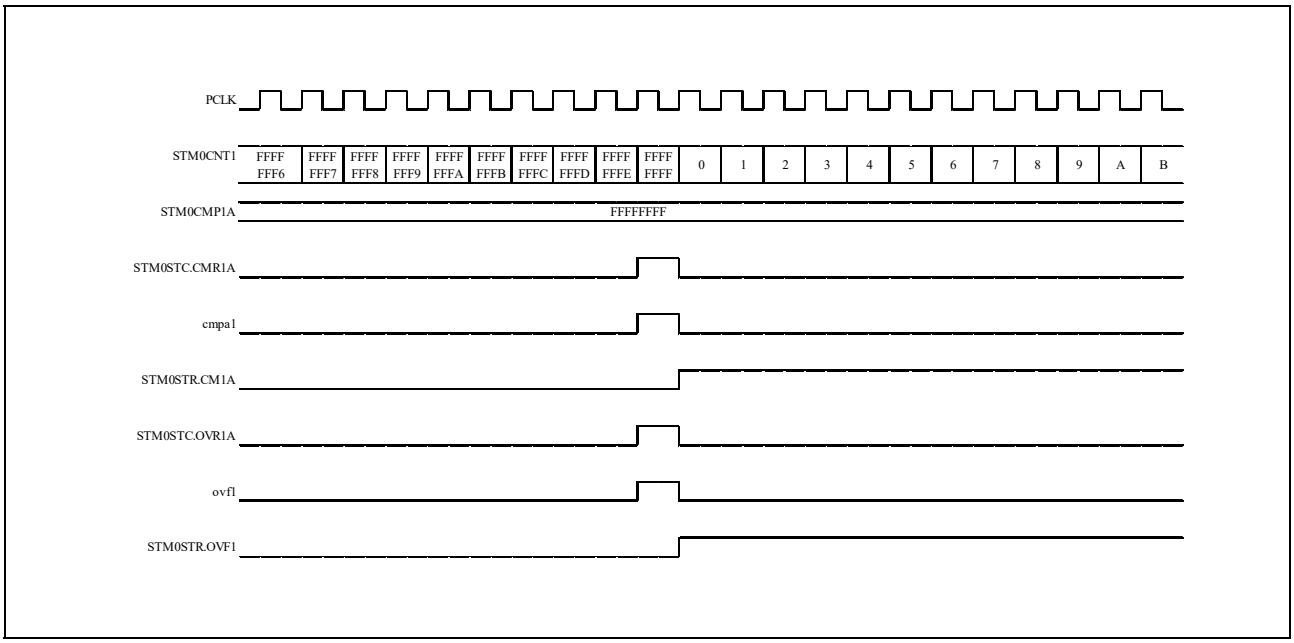



**Figure 24.6 Compare match event conflicting with its clear**

#### 24.4.3.3 Confliction of flag (compare match/overflow) set with counter stop

When the set signal of the flag conflict with the counter stop ,the flag will not be set until the coming of next count source clock after restart of the count. To compare match flag, if you update the compare match register or counter register with other values within the stop of the counter register, the flag will not be set even though the coming of next count source clock after restart of the count.To the overflow flag, if you update the counter register with other values within its stop, the flag will not be set even though the coming of next count source clock after restart of the count. In the figure below, the compare match flat will be set when the counter restarted if you had not updated the compare match register but if you had updated the compare match register during counter stop the compare match flat will not be set when the counter restarted.

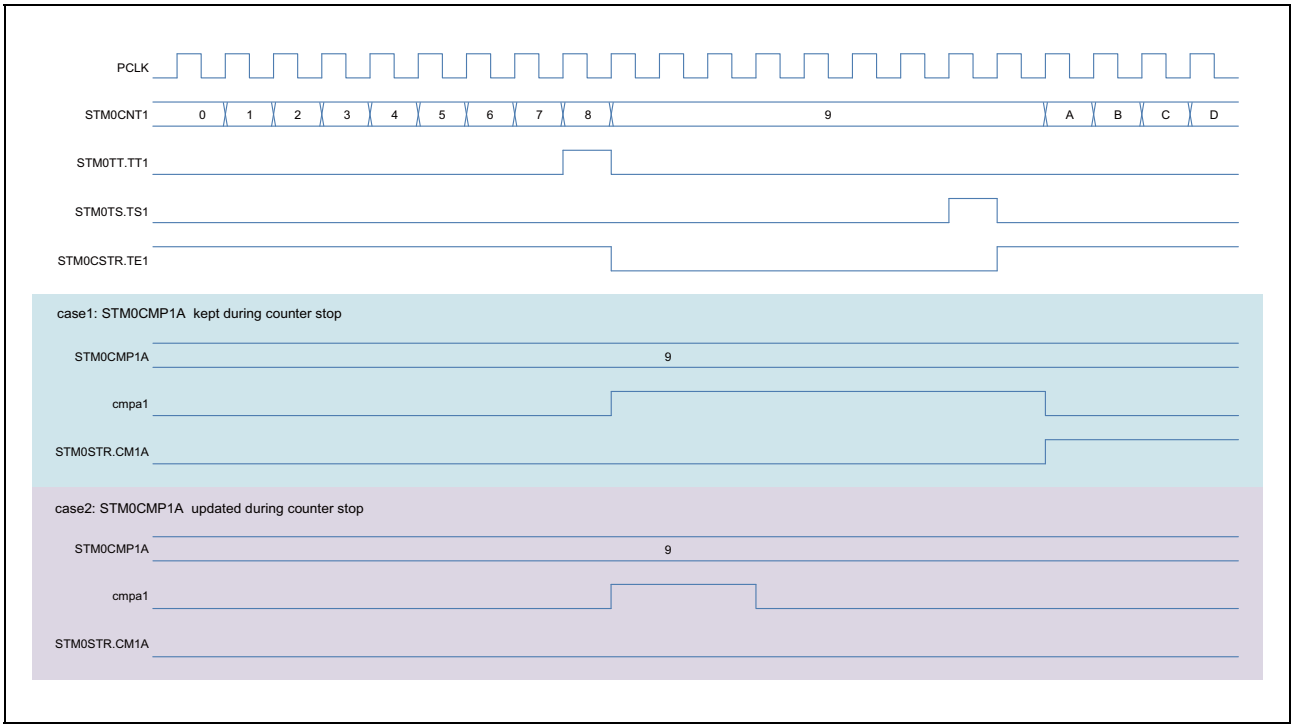



**Figure 24.7 Compare match event conflicting with counter stop**

### 24.4.3.4 Confliction of flag set with emulation mode start

When the set signal of the flag conflict with emulation mode, the flag will not be set.And when restore from the emulation mode it would be check once again whether the flag should be set or not.

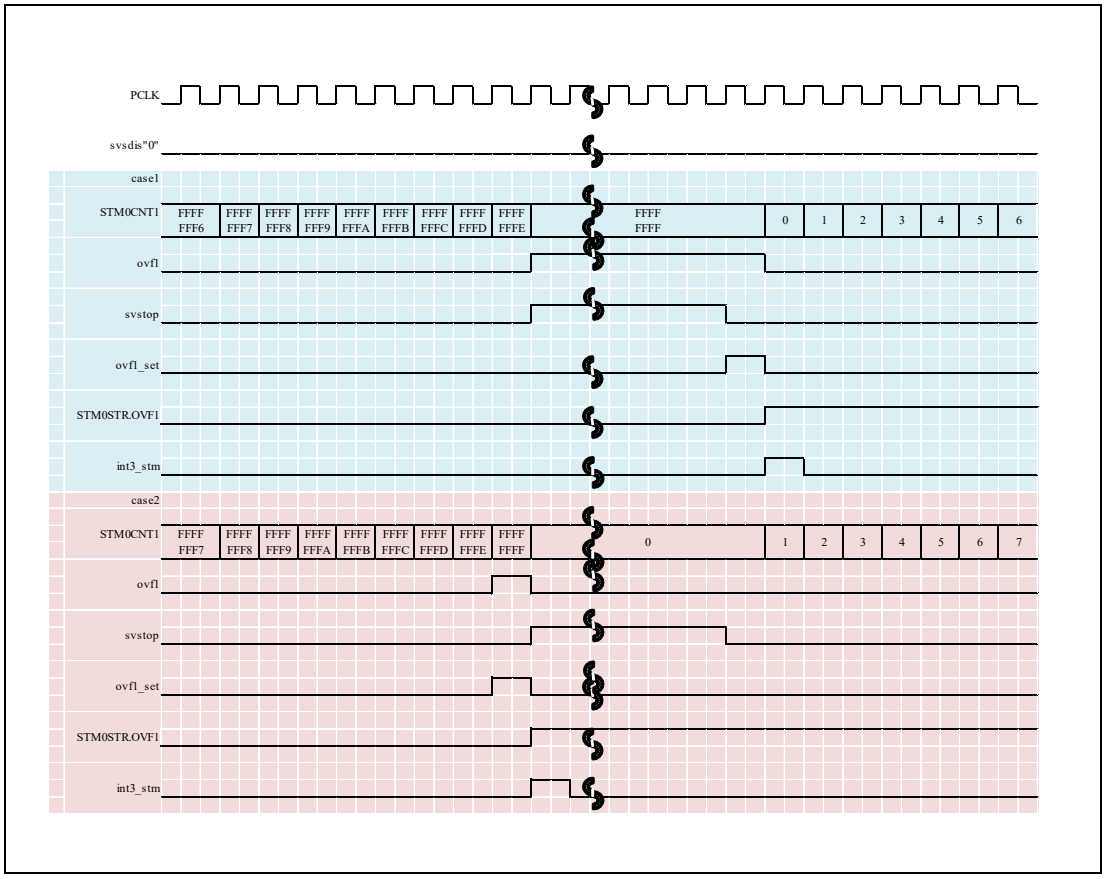



**Figure 24.8 Overflow event conflicting with emulation mode**

### 24.4.3.5 Confliction of interrupt requirement with flag clear

When the flag has been set in STM0STR, the interrupt requirement according to the corresponding flag set event will not be sent out,but only in the case that the corresponding flag clear happens at the same time .And in this case not only the interrupt requirement will be sent out but also the flag will not be cleared.

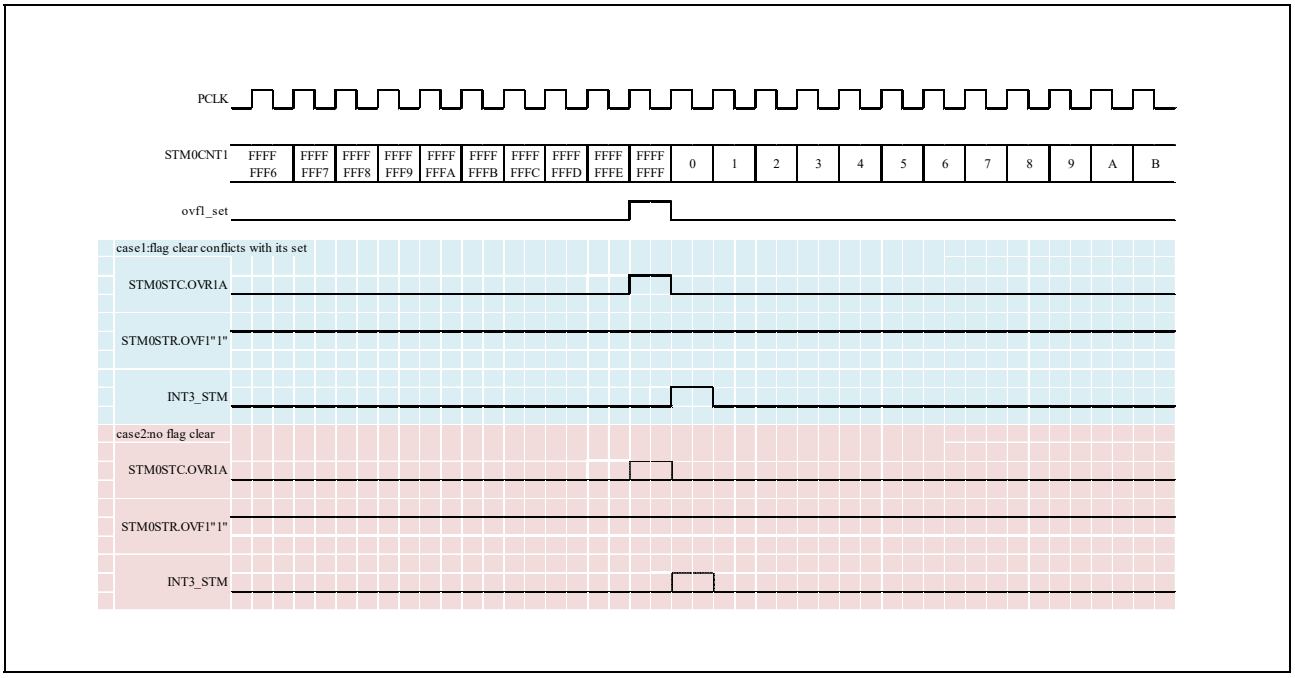



**Figure 24.9 Interrupt requirement conflicting with flag clear**

### 24.4.4 Access to register

Including the registers of STM0STR, STM0STC, all the counter registers and compare match registers should be access by 32-bit only.

The bit width of compare match register and the counter register in CNT0 is 64-bit. To those register, 64-bit access is supported and the 64-bit access will be divided into twice accesses of the 32-bit peripheral bus which used in STM. In case of individual 32-bit write or read accesses to 64-bit counter or compare registers of CNT0, the upper 32-bit register will be processed after the lower 32-bit register. Writing access to a 64-bit register, the register value will be updated when the upper 32-bit register is processed while reading access to a 64-bit register, the value of the register at that moment will be read out.

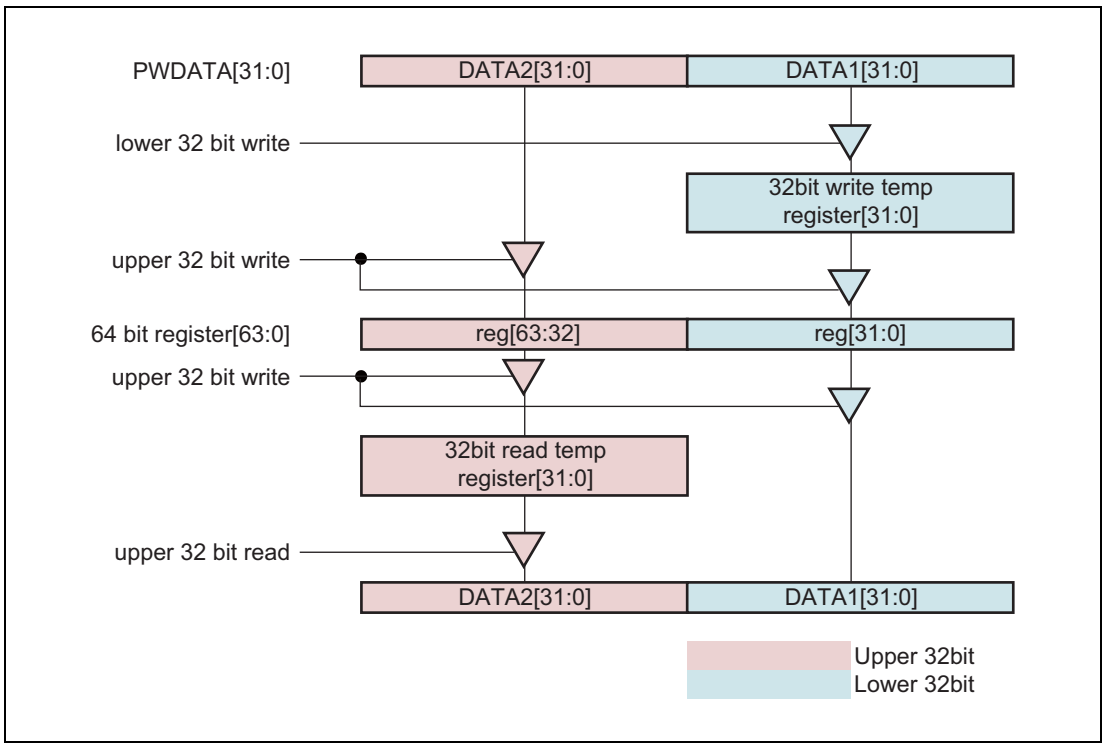



**Figure 24.10 Access to register**

### 24.4.5 Resets

There are Application reset and System reset in STM. To System reset, all of the STM registers will be initialized, to the Application reset, you can mask it or not. All of the registers behave as shown in the below table upon the Application reset or System reset.

**Table 24.24 Reset**

| Registers or bits in STM | | Application reset | | System reset |
|---|---|---|---|---|
| | | Masked (RMm = 1) | UnMask (RMm = 0) | |
| Counters | Counters in CNTm | KEPT | INIT | INIT |
| | Counters in other channels | INIT | INIT | INIT |
| Compare match register | Compare match register | INIT | INIT | INIT |
| STMnTS | TSm | KEPT | INIT | INIT |
| | Other TS bits in STMnTS[3:0] | INIT | INIT | INIT |
| STMnTT | TTm | KEPT | INIT | INIT |
| | Other TT bits in STMnTT[3:0] | INIT | INIT | INIT |
| STMnCSTR | Tem | KEPT | INIT | INIT |
| | other TE bits in STMnCSTR[3:0] | INIT | INIT | INIT |
| STMnSTR | OVFm | KEPT | INIT | INIT |
| | Other OVF or CM bits in STMnSTR[19:0] | INIT | INIT | INIT |
| STMnSTC | STMnSTC[19:0] | INIT | INIT | INIT |
| STMnRM | RMm | KEPT | KEPT | INIT |
| | Other RM bits in STMnRM[3:0] | INIT | INIT | INIT |
| STMnCKSEL | STMnCKSEL[3:0] | INIT | INIT | INIT |
| STMnIS | STMnIS[15:0] | INIT | INIT | INIT |

Note 1. m: m = 0 to 3
Note 2. INIT: initialized
Note 3. KEPT: values are kept

## 24.5 Difference among P1M-C, P1H-C and P1H-CE

There are differences in channels, interrupts, and external pins. For details, see **Table 24.1**, and **Table 24.5**.

# Section 25 Generic Timer Module (GTM)

Generic timer module (GTM) is a modular timer unit used to support chassis control applications. It is designed to unload the CPU from a high interrupt load. Most of the task of GTM can run independently from the CPU.

## 25.1 Feature

### 25.1.1 IP version

The table below shows IP version in each device.

**Table 25.1 IP version**

| IP | P1M-C (Config 2) | P1H-C, P1H-CE (Config 3) |
|---|---|---|
| GTM-IP | GTM-IP 207 | GTM-IP 208 |

### 25.1.2 Number of Sub-Units and Channels

The table below shows the availability of the GTM.

**Table 25.2 Sub-Units and Channels**

| | P1M-C | | | P1H-C, P1H-CE | | |
|---|---|---|---|---|---|---|
| Submodule | Number of Instances | Instance name | Channels | Number of Instances | Instance name | Channels |
| ARU | 1 | — | — | 1 | — | — |
| CMU | 1 | — | 8 (n = 0 to 7) | 1 | — | 8 (n = 0 to 7) |
| TBU | 1 | — | 2 (n = 0, 1) | 1 | — | 2 (n = 0, 1) |
| TIM | 2 | TIM0,<br>TIM1 | 8 (TIM0: n = 0 to 7)<br>8 (TIM1: n = 0 to 7) | 2 | TIM0,<br>TIM1 | 8 (TIM0: n = 0 to 7)<br>8 (TIM1: n = 0 to 7) |
| ATOM | 2 | ATOM0,<br>ATOM1 | 8 (ATOM0: n = 0 to 7)<br>4 (ATOM1: n = 0 to 3) | 3 | ATOM0,<br>ATOM1*2,<br>ATOM2*3 | 8 (ATOM0: n = 0 to 7)<br>8 (ATOM1: n = 0 to 7)<br>5 (ATOM2: n = 0 to 4) |
| DTM | 2 | DTM24,<br>DTM26 | 4 (DTM24: n = 0 to 3)<br>4 (DTM26: n = 0 to 3) | 3 | DTM24,<br>DTM26,<br>DTM28 | 4 (DTM24: n = 0 to 3)<br>4 (DTM26: n = 0 to 3)<br>4 (DTM28: n = 0 to 3) |
| MCS*1 | 1 | MCS0 | 9 (MCS0: n = 0 to 8) | 2 | MCS0,<br>MCS1 | 9 (MCS0: n = 0 to 8)<br>6 (MCS1: n = 0 to 5) |
| MCFG | — | — | — | 1 | — | — |
| ICM | 1 | — | — | 1 | — | — |
| CMP | 1 | — | — | 1 | — | — |
| MON | 1 | — | — | 1 | — | — |

Note 1. Each instance has a memory of 2x256 32-bit words.

Note 2. P1H-C (4MB, BGA-156) does not support ATOM1 (Ch. 4 to 7) output terminal, but internal functions are available.

Note 3. P1H-C (4MB, BGA-156) does not support ATOM2 all channels output terminal, but internal functions are available.

### 25.1.3 Register/RAM Base Address

GTM base addresses are listed in the following table. GTM register and RAM addresses are given as offsets from the base addresses in general.

**Table 25.3 Register and RAM Base Addresses**

| Base Address Name | Base Address |
|---|---|
| <GTM0_base> | FFE0 $0000_H$ |

### 25.1.4 Clock Supply

Clock supply by and to GTM is listed in the following table.

**Table 25.4 Clock Supply**

| Unit Name | Clock for the Unit | Supply Clock Name |
|---|---|---|
| GTM | AEI-bus clock (AEI_CLK) | CLK_HSB |
| | GTM global clock ((AEI_)SYS_CLK) | CLK_HSB |

### 25.1.5 Interrupt and DMA/DTS Requests

GTM interrupt requests are listed in the following table.

**Table 25.5 Interrupt Requests (1/2)**

| Interrupt Name | Description | Interrupt Number | DMA Trigger Number | DTS Trigger Number | Supporting Device | |
|---|---|---|---|---|---|---|
| | | | | | P1M-C | P1H-C, P1H-CE |
| INTGTM0ERR | GTM Error Interrupt | 47 | — | — | √ | √ |
| INTGTM0AEI | AEI Shared interrupt | 48 | — | — | √ | √ |
| INTGTM0ARU0 | ARU_NEW_DATA0 interrupt | 49 | 10 | — | √ | √ |
| INTGTM0ARU1 | ARU_NEW_DATA1 interrupt | 50 | 11 | — | √ | √ |
| INTGTM0ARU2 | ARU_ACC_ACK interrupt | 51 | 12 | — | √ | √ |
| INTGTM0CMP | CMP Shared interrupt | 52 | 13 | — | √ | √ |
| INTGTM0TIM00*1 | TIM0 Shared interrupts (TIM0_IRQ0) | 53 | 14 | 10 | √ | √ |
| INTGTM0TIM01*1 | TIM0 Shared interrupts (TIM0_IRQ1) | 141 | 15 | 11 | √ | √ |
| INTGTM0TIM02*1 | TIM0 Shared interrupts (TIM0_IRQ2) | 54 | 16 | 12 | √ | √ |
| INTGTM0TIM03*1 | TIM0 Shared interrupts (TIM0_IRQ3) | 142 | 17 | 13 | √ | √ |
| INTGTM0TIM04*1 | TIM0 Shared interrupts (TIM0_IRQ4) | 55 | 18 | 14 | √ | √ |
| INTGTM0TIM05*1 | TIM0 Shared interrupts (TIM0_IRQ5) | 143 | 19 | 15 | √ | √ |
| INTGTM0TIM06*1 | TIM0 Shared interrupts (TIM0_IRQ6) | 56 | 20 | 16 | √ | √ |
| INTGTM0TIM07*1 | TIM0 Shared interrupts (TIM0_IRQ7) | 144 | 21 | 17 | √ | √ |
| INTGTM0TIM10*1 | TIM1 Shared interrupts (TIM1_IRQ0) | 57 | 22 | 18 | √ | √ |
| INTGTM0TIM11*1 | TIM1 Shared interrupts (TIM1_IRQ1) | 145 | 23 | 19 | √ | √ |
| INTGTM0TIM12*1 | TIM1 Shared interrupts (TIM1_IRQ2) | 58 | 24 | 20 | √ | √ |
| INTGTM0TIM13*1 | TIM1 Shared interrupts (TIM1_IRQ3) | 146 | 25 | 21 | √ | √ |
| INTGTM0TIM14*1 | TIM1 Shared interrupts (TIM1_IRQ4) | 59 | 26 | 22 | √ | √ |
| INTGTM0TIM15*1 | TIM1 Shared interrupts (TIM1_IRQ5) | 147 | 27 | 23 | √ | √ |
| INTGTM0TIM16*1 | TIM1 Shared interrupts (TIM1_IRQ6) | 60 | 28 | 24 | √ | √ |
| INTGTM0TIM17*1 | TIM1 Shared interrupts (TIM1_IRQ7) | 148 | 29 | 25 | √ | √ |
| INTGTM0MCS00*1 | MCS0 Interrupt for channel (MCS0_IRQ0) | 61 | 30 | 118 | √ | √ |
| INTGTM0MCS01*1 | MCS0 Interrupt for channel (MCS0_IRQ1) | 149 | 31 | 119 | √ | √ |
| INTGTM0MCS02*1 | MCS0 Interrupt for channel (MCS0_IRQ2) | 62 | 32 | 120 | √ | √ |
| INTGTM0MCS03*1 | MCS0 Interrupt for channel (MCS0_IRQ3) | 150 | 33 | 121 | √ | √ |
| INTGTM0MCS04*1 | MCS0 Interrupt for channel (MCS0_IRQ4) | 63 | 34 | 122 | √ | √ |
| INTGTM0MCS05*1 | MCS0 Interrupt for channel (MCS0_IRQ5) | 151 | 35 | 123 | √ | √ |
| INTGTM0MCS06*1 | MCS0 Interrupt for channel (MCS0_IRQ6) | 64 | 36 | 124 | √ | √ |
| INTGTM0MCS07*1 | MCS0 Interrupt for channel (MCS0_IRQ7) | 152 | 37 | 125 | √ | √ |
| INTGTM0MCS08*1 | MCS0 Interrupt for channel (MCS0_IRQ8) | 65 | 38 | 126 | √ | √ |
| INTGTM0MCS10*1 | MCS1 Interrupt for channel (MCS1_IRQ0) | 153 | 39 | 100*2 | — | √ |
| INTGTM0MCS11*1 | MCS1 Interrupt for channel (MCS1_IRQ1) | 66 | 40 | 101*2 | — | √ |
| INTGTM0MCS12*1 | MCS1 Interrupt for channel (MCS1_IRQ2) | 154 | 41 | 102*2 | — | √ |
| INTGTM0MCS13*1 | MCS1 Interrupt for channel (MCS1_IRQ3) | 67 | 42 | 103*2 | — | √ |
| INTGTM0MCS14*1 | MCS1 Interrupt for channel (MCS1_IRQ4) | 155 | 43 | 104*2 | — | √ |
| INTGTM0MCS15*1 | MCS1 Interrupt for channel (MCS1_IRQ5) | 68 | 44 | 105*2 | — | √ |
| INTGTM0ATOM00*1 | ATOM0 Shared interrupts (ATOM0_IRQ0) | 69 | 45 | 127 | √ | √ |
| INTGTM0ATOM01*1 | ATOM0 Shared interrupts (ATOM0_IRQ1) | 156 | 46 | 118*2 | √ | √ |

Table 25.5 Interrupt Requests (2/2)

| Interrupt Name | Description | Interrupt Number | DMA Trigger Number | DTS Trigger Number | Supporting Device | |
|---|---|---|---|---|---|---|
| | | | | | P1M-C | P1H-C, P1H-CE |
| INTGTM0ATOM02*1 | ATOM0 Shared interrupts (ATOM0_IRQ2) | 70 | 47 | 119*2 | √ | √ |
| INTGTM0ATOM03*1 | ATOM0 Shared interrupts (ATOM0_IRQ3) | 157 | 48 | 120*2 | √ | √ |
| INTGTM0ATOM04*1 | ATOM0 Shared interrupts (ATOM0_IRQ4) | 71 | 49 | 121*2 | √ | √ |
| INTGTM0ATOM05*1 | ATOM0 Shared interrupts (ATOM0_IRQ5) | 158 | 50 | 122*2 | √ | √ |
| INTGTM0ATOM06*1 | ATOM0 Shared interrupts (ATOM0_IRQ6) | 72 | 51 | 123*2 | √ | √ |
| INTGTM0ATOM07*1 | ATOM0 Shared interrupts (ATOM0_IRQ7) | 159 | 52 | 124*2 | √ | √ |
| INTGTM0ATOM10*1 | ATOM1 Shared interrupts (ATOM1_IRQ0) | 73 | 53 | 125*2 | √ | √ |
| INTGTM0ATOM11*1 | ATOM1 Shared interrupts (ATOM1_IRQ1) | 160 | 54 | 126*2 | √ | √ |
| INTGTM0ATOM12*1 | ATOM1 Shared interrupts (ATOM1_IRQ2) | 74 | 55 | 127*2 | √ | √ |
| INTGTM0ATOM13*1 | ATOM1 Shared interrupts (ATOM1_IRQ3) | 161 | 56 | 96*2 | √ | √ |
| INTGTM0ATOM14*1 | ATOM1 Shared interrupts (ATOM1_IRQ4) | 75 | 57 | 97*2 | — | √ |
| INTGTM0ATOM15*1 | ATOM1 Shared interrupts (ATOM1_IRQ5) | 162 | 58 | 98*2 | — | √ |
| INTGTM0ATOM16*1 | ATOM1 Shared interrupts (ATOM1_IRQ6) | 76 | 59 | 99*2 | — | √ |
| INTGTM0ATOM17*1 | ATOM1 Shared interrupts (ATOM1_IRQ7) | 163 | 60 | 80*2 | — | √ |
| INTGTM0ATOM20*1 | ATOM2 Shared interrupts (ATOM2_IRQ0) | 77 | 61 | 81*2 | — | √ |
| INTGTM0ATOM21*1 | ATOM2 Shared interrupts (ATOM2_IRQ1) | 164 | 62 | 82*2 | — | √ |
| INTGTM0ATOM22*1 | ATOM2 Shared interrupts (ATOM2_IRQ2) | 78 | 63 | 83*2 | — | √ |
| INTGTM0ATOM23*1 | ATOM2 Shared interrupts (ATOM2_IRQ3) | 165 | 64 | 66*2 | — | √ |
| INTGTM0ATOM24*1 | ATOM2 Shared interrupts (ATOM2_IRQ4) | 79 | 65 | 67*2 | — | √ |

Note 1. These interrupts are also connected to PIC.

Note 2. Only available when secondary DTS trigger is selected.

### 25.1.6 External Input and Output Pins

External input/output pins of GTM are listed below.

**Table 25.6 Pin Function Information**

| Pin Name | I/O | Description | Supporting Device | |
|---|---|---|---|---|
| | | | P1M-C, P1H-C (4MB, BGA-156) | P1H-C (4MB, BGA-292), P1H-C (8MB), P1H-CE |
| GTM0I0 | I | Timer input signals for TIM0 (via PIC) | √ | √ |
| GTM0I1 | | | √ | √ |
| GTM0I2 | | | √ | √ |
| GTM0I3 | | | √ | √ |
| GTM0I4 | | | √ | √ |
| GTM0I5 | | | √ | √ |
| GTM0I6 | | | √ | √ |
| GTM0I7 | | | √ | √ |
| GTM1I0 | I | Timer input signals for TIM1 (via PIC) | √ | √ |
| GTM1I1 | | | √ | √ |
| GTM1I2 | | | √ | √ |
| GTM1I3 | | | √ | √ |
| GTM1I4 | | | √ | √ |
| GTM1I5 | | | √ | √ |
| GTM1I6 | | | √ | √ |
| GTM1I7 | | | √ | √ |
| GTMAT0O0*1 | O | Timer output signals for ATOM0 | √ | √ |
| GTMAT0O1*1 | | | √ | √ |
| GTMAT0O2*1 | | | √ | √ |
| GTMAT0O3*1 | | | √ | √ |
| GTMAT0O4 | | | √ | √ |
| GTMAT0O5 | | | √ | √ |
| GTMAT0O6 | | | √ | √ |
| GTMAT0O7 | | | √ | √ |
| GTMAT1O0*1 | O | Timer output signals for ATOM1 | √ | √ |
| GTMAT1O1*1 | | | √ | √ |
| GTMAT1O2*1 | | | √ | √ |
| GTMAT1O3*1 | | | √ | √ |
| GTMAT1O4 | | | — | √ |
| GTMAT1O5 | | | — | √ |
| GTMAT1O6 | | | — | √ |
| GTMAT1O7 | | | — | √ |
| GTMAT2O0*1 | O | Timer output signals for ATOM2 | — | √ |
| GTMAT2O1*1 | | | — | √ |
| GTMAT2O2*1 | | | — | √ |
| GTMAT2O3*1 | | | — | √ |
| GTMAT2O4 | | | — | √ |
| GTMAT0O0N*1 | O | Inverted timer output signals for ATOM0 | √ | √ |
| GTMAT0O1N*1 | | | √ | √ |
| GTMAT0O2N*1 | | | √ | √ |
| GTMAT0O3N*1 | | | √ | √ |
| GTMAT1O0N*1 | O | Inverted timer output signals for ATOM1 | √ | √ |
| GTMAT1O1N*1 | | | √ | √ |
| GTMAT1O2N*1 | | | √ | √ |
| GTMAT1O3N*1 | | | √ | √ |
| GTMAT2O0N*1 | O | Inverted timer output signals for ATOM2 | — | √ |
| GTMAT2O1N*1 | | | — | √ |
| GTMAT2O2N*1 | | | — | √ |
| GTMAT2O3N*1 | | | — | √ |
| ESO0Z | I | Hi-Z control for timer output via PIC | √ | √ |
| ESO1Z | | | √ | √ |
| ESO2Z | | | — | √ |

Note 1. Output buffer of these signals could be controlled to Hi-Z by PIC.

## 25.2 Overview

### 25.2.1 Functional Overview

GTM is a modular timer unit and consists at least of the following submodules.

- Advanced Routing Unit (ARU)
- Clock Management Unit (CMU)
- Time Base Unit (TBU)
- Timer Input Module (TIM)
- ARU-connected Timer Output Module (ATOM)
- Multi-Channel Sequencer (MCS)
- Interrupt Concentrator Module (ICM)
- Output Compare Unit (CMP)
- Monitoring Unit (MON)
- Dead Time Module (DTM)
- Memory Configuration Module (MCFG; Available only in P1H-C, and P1H-CE)

## 25.2.2 Block Diagram

The following figure shows the block diagram.

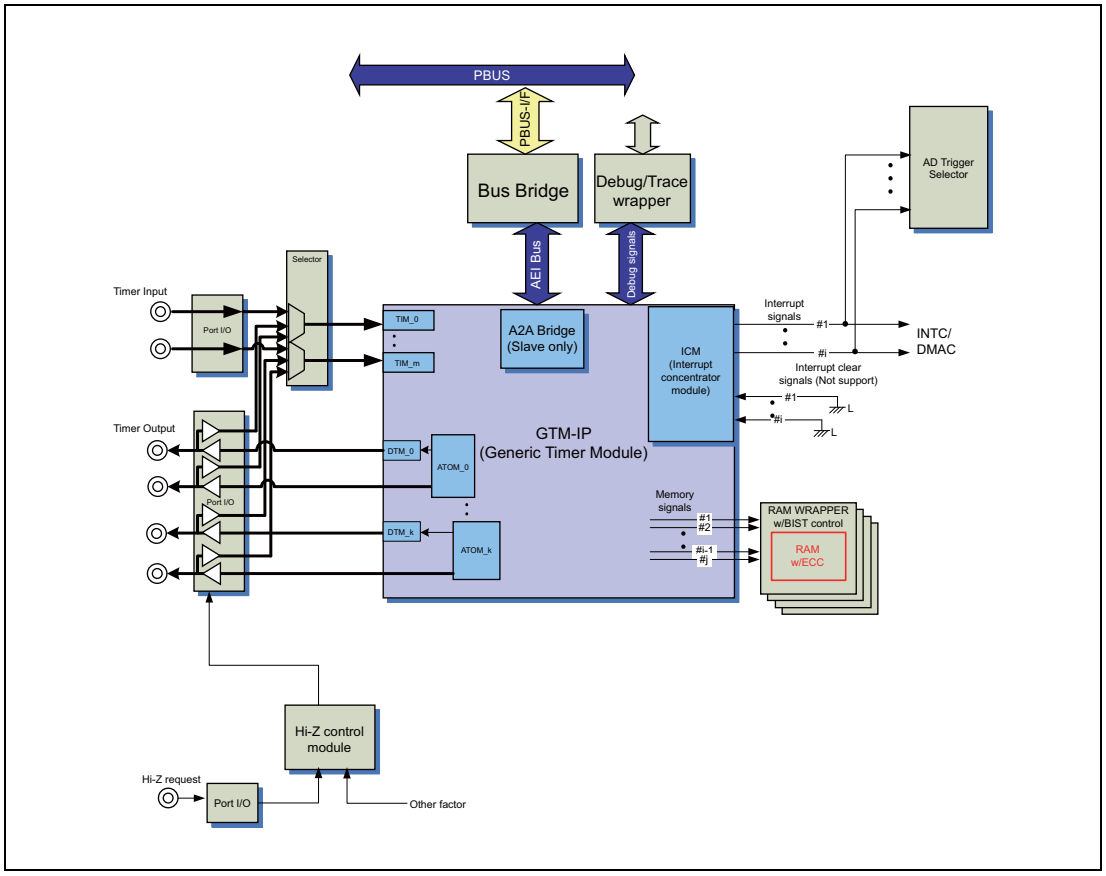



**Figure 25.1 GTM integration diagram**

# 25.3 Registers

## 25.3.1 List of Registers

GTM registers are listed in the following table.

**Table 25.7 List of Registers (1/37)**

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + $00000_H$ | GTM0GTMREV | GTM-IP Version control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00004_H$ | GTM0GTMRST | GTM-IP Global reset register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00008_H$ | GTM0GTMCTRL | GTM-IP Global control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0000C_H$ | GTM0GTMAEIADDRXPT | GTM-IP AEI Timeout exception address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00010_H$ | GTM0GTMIRQNOTIFY | GTM-IP Interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00014_H$ | GTM0GTMIRQEN | GTM-IP Interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00018_H$ | GTM0GTMIRQFORCINT | GTM-IP Software interrupt generation register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0001C_H$ | GTM0GTMIRQMODE | GTM-IP top level interrupts mode selection. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00020_H$ | GTM0GTMEIRQEN | GTM-IP Error interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00024_H$ | GTM0GTMHWCONF | GTM-IP Hardware Configuration | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00030_H$ | GTM0GTMBRIDGEMODE | The operation mode for the AEI bridge | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00034_H$ | GTM0GTMBRIDGEPTR1 | AEI bridge status pointer 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00038_H$ | GTM0GTMBRIDGEPTR2 | AEI bridge status pointer 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00040_H$ | GTM0GTMTIM0AUXINSRC | GTM-IP TIM0 module AUX_IN source selection register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00044_H$ | GTM0GTMTIM1AUXINSRC | GTM-IP TIM1 module AUX_IN source selection register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0005C_H$ | GTM0GTMEXTCAPEN0 | GTM-IP trigger event forwarding in from TIM0 and TIM1 to MCS 0. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00060_H$ | GTM0GTMEXTCAPEN1 | GTM-IP trigger event forwarding in from TIM1 to MCS1. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00098_H$ | GTM0GTMATOM0OUT | GTM-IP ATOM output level | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0009C_H$ | GTM0GTMATOM2OUT | GTM-IP ATOM output level | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00100_H$ | GTM0TBUCHEN | TBU global channel enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00104_H$ | GTM0TBU0CTRL | TBU channel 0 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00108_H$ | GTM0TBU0BASE | TBU channel 0 base | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0010C_H$ | GTM0TBU1CTRL | TBU channel 1 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00110_H$ | GTM0TBU1BASE | TBU channel 1 base | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (2/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + $00180_H$ | GTM0MONSTATUS | Monitor Status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM | <GTM0_base> + $00184_H$ | GTM0MONACTIVITY0 | Monitor activity register 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0018C_H$ | GTM0MONACTIVITYMCS0 | Monitor activity register for MCS0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00190_H$ | GTM0MONACTIVITYMCS1 | Monitor activity register for MCS1 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00200_H$ | GTM0CMPEN | Comparator enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00204_H$ | GTM0CMPIRQNOTIFY | Event notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00208_H$ | GTM0CMPIRQEN | Interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0020C_H$ | GTM0CMPIRQFORCINT | Interrupt force register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00210_H$ | GTM0CMPIRQMODE | IRQ mode configuration register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00214_H$ | GTM0CMPEIRQEN | Error interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00280_H$ | GTM0ARUACCESS | ARU access register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00284_H$ | GTM0ARUDATAH | ARU access register upper data word | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00288_H$ | GTM0ARUDATAL | ARU access register lower data word | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0028C_H$ | GTM0ARUDBGACCESS0 | Debug access channel 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00290_H$ | GTM0ARUDBGDATA0H | Debug access 0 transfer register upper data word | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00294_H$ | GTM0ARUDBGDATA0L | Debug access 0 transfer register lower data word | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00298_H$ | GTM0ARUDBGACCESS1 | Debug access channel 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0029C_H$ | GTM0ARUDBGDATA1H | Debug access 1 transfer register upper data word | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $002A0_H$ | GTM0ARUDBGDATA1L | Debug access 1 transfer register lower data word | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $002A4_H$ | GTM0ARUIRQNOTIFY | ARU Interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $002A8_H$ | GTM0ARUIRQEN | ARU Interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $002AC_H$ | GTM0ARUIRQFORCINT | Register for forcing the *ARU_NEW_DATA*_IRQ interrupt | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $002B0_H$ | GTM0ARUIRQMODE | IRQ mode configuration register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $002B4_H$ | GTM0ARUCADDREND | ARU caddr counter end value | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00300_H$ | GTM0CMUCLKEN | Clock enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00304_H$ | GTM0CMUGCLKNUM | Global clock control numerator | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $00308_H$ | GTM0CMUGCLKDEN | Global clock control denominator | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (3/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 0030C$_H$ | GTM0CMUCLK0CTRL | Control for clock source 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00310$_H$ | GTM0CMUCLK1CTRL | Control for clock source 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00314$_H$ | GTM0CMUCLK2CTRL | Control for clock source 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00318$_H$ | GTM0CMUCLK3CTRL | Control for clock source 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0031C$_H$ | GTM0CMUCLK4CTRL | Control for clock source 4 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00320$_H$ | GTM0CMUCLK5CTRL | Control for clock source 5 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00324$_H$ | GTM0CMUCLK6CTRL | Control for clock source 6 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00328$_H$ | GTM0CMUCLK7CTRL | Control for clock source 7 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00348$_H$ | GTM0CMUGLBCTRL | Synchronizing ARU and clock source | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00600$_H$ | GTM0ICMIRQG0 | ICM Interrupt group register covering infra-structural and safety components (ARU, AEI, CMP) | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00608$_H$ | GTM0ICMIRQG2 | ICM Interrupt group register covering TIM0, TIM1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00610$_H$ | GTM0ICMIRQG4 | ICM Interrupt group register covering MCS0 to MCS1 submodules | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00624$_H$ | GTM0ICMIRQG9 | ICM Interrupt group register covering GTM-IP output submodules ATOM0, ATOM1, ATOM2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00630$_H$ | GTM0ICMIRQGMEI | ICM Interrupt group register for module error interrupt information | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00638$_H$ | GTM0ICMIRQGCEI1 | ICM Interrupt group register 1 for channel error interrupt information | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00640$_H$ | GTM0ICMIRQGCEI3 | ICM Interrupt group register 3 for channel error interrupt information | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00648$_H$ | GTM0ICMIRQGMCS0CI | ICM Interrupt group MCS 0 for Channel Interrupt information | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0064C$_H$ | GTM0ICMIRQGMCS1CI | ICM Interrupt group MCS 0 for Channel Error Interrupt information | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00664$_H$ | GTM0ICMIRQGMCS0CEI | ICM Interrupt group MCS 1 for Channel Interrupt information | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00668$_H$ | GTM0ICMIRQGMCS1CEI | ICM Interrupt group MCS 1 for Channel Error Interrupt information | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 00F40$_H$ | GTM0MCFGCTRL | Memory layout configuration. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01000$_H$ | GTM0TIM00GPR0 | channel 0 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01004$_H$ | GTM0TIM00GPR1 | channel 0 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01008$_H$ | GTM0TIM00CNT | channel 0 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0100C$_H$ | GTM0TIM00ECNT | channel 0 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01010$_H$ | GTM0TIM00CNTS | channel 0 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (4/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + $01014_H$ | GTM0TIM00TDUC | channel 0 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01018_H$ | GTM0TIM00TDUV | channel 0 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0101C_H$ | GTM0TIM00FLTRE | channel 0 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01020_H$ | GTM0TIM00FLTFE | channel 0 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01024_H$ | GTM0TIM00CTRL | channel 0 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01028_H$ | GTM0TIM00ECTRL | channel 0 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0102C_H$ | GTM0TIM00IRQNOTIFY | channel 0 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01030_H$ | GTM0TIM00IRQEN | channel 0 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01034_H$ | GTM0TIM00IRQFORCINT | channel 0 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01038_H$ | GTM0TIM00IRQMODE | IRQ mode configuration register 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0103C_H$ | GTM0TIM00EIRQEN | channel 0 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01074_H$ | GTM0TIM0INPVAL | TIM input value observation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01078_H$ | GTM0TIM0INSRC | TIM AUX IN source selection | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0107C_H$ | GTM0TIM0RST | TIM global software reset | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01080_H$ | GTM0TIM01GPR0 | channel 0 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01084_H$ | GTM0TIM01GPR1 | channel 0 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01088_H$ | GTM0TIM01CNT | channel 0 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0108C_H$ | GTM0TIM01ECNT | channel 0 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01090_H$ | GTM0TIM01CNTS | channel 1 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01094_H$ | GTM0TIM01TDUC | channel 1 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01098_H$ | GTM0TIM01TDUV | channel 1 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0109C_H$ | GTM0TIM01FLTRE | channel 1 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $010A0_H$ | GTM0TIM01FLTFE | channel 1 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $010A4_H$ | GTM0TIM01CTRL | channel 1 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $010A8_H$ | GTM0TIM01ECTRL | channel 1 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $010AC_H$ | GTM0TIM01IRQNOTIFY | channel 1 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $010B0_H$ | GTM0TIM01IRQEN | channel 1 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (5/37)

| Module name | Address | Symbol | Register Name | Access Width | Support 207 | Support 208 | Access Protection PBG | Access Protection Other |
|---|---|---|---|---|---|---|---|---|
| GTM0 | <GTM0_base> + $010B4_H$ | GTM0TIM01IRQFORCINT | channel 1 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $010B8_H$ | GTM0TIM01IRQMODE | IRQ mode configuration register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $010BC_H$ | GTM0TIM01EIRQEN | channel 1 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01100_H$ | GTM0TIM02GPR0 | channel 2 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01104_H$ | GTM0TIM02GPR1 | channel 2 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01108_H$ | GTM0TIM02CNT | channel 2 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0110C_H$ | GTM0TIM02ECNT | channel 2 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01110_H$ | GTM0TIM02CNTS | channel 2 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01114_H$ | GTM0TIM02TDUC | channel 2 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01118_H$ | GTM0TIM02TDUV | channel 2 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0111C_H$ | GTM0TIM02FLTRE | channel 2 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01120_H$ | GTM0TIM02FLTFE | channel 2 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01124_H$ | GTM0TIM02CTRL | channel 2 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01128_H$ | GTM0TIM02ECTRL | channel 2 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0112C_H$ | GTM0TIM02IRQNOTIFY | channel 2 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01130_H$ | GTM0TIM02IRQEN | channel 2 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01134_H$ | GTM0TIM02IRQFORCINT | channel 2 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01138_H$ | GTM0TIM02IRQMODE | IRQ mode configuration register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0113C_H$ | GTM0TIM02EIRQEN | channel 2 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01180_H$ | GTM0TIM03GPR0 | channel 3 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01184_H$ | GTM0TIM03GPR1 | channel 3 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01188_H$ | GTM0TIM03CNT | channel 3 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0118C_H$ | GTM0TIM03ECNT | channel 3 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01190_H$ | GTM0TIM03CNTS | channel 3 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01194_H$ | GTM0TIM03TDUC | channel 3 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01198_H$ | GTM0TIM03TDUV | channel 3 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0119C_H$ | GTM0TIM03FLTRE | channel 3 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (6/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 011A0$_H$ | GTM0TIM03FLTFE | channel 3 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 011A4$_H$ | GTM0TIM03CTRL | channel 3 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 011A8$_H$ | GTM0TIM03ECTRL | channel 3 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 011AC$_H$ | GTM0TIM03IRQNOTIFY | channel 3 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 011B0$_H$ | GTM0TIM03IRQEN | channel 3 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 011B4$_H$ | GTM0TIM03IRQFORCINT | channel 3 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 011B8$_H$ | GTM0TIM03IRQMODE | IRQ mode configuration register 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 011BC$_H$ | GTM0TIM03EIRQEN | channel 3 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01200$_H$ | GTM0TIM04GPR0 | channel 4 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01204$_H$ | GTM0TIM04GPR1 | channel 4 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01208$_H$ | GTM0TIM04CNT | channel 4 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0120C$_H$ | GTM0TIM04ECNT | channel 4 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01210$_H$ | GTM0TIM04CNTS | channel 4 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01214$_H$ | GTM0TIM04TDUC | channel 4 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01218$_H$ | GTM0TIM04TDUV | channel 4 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0121C$_H$ | GTM0TIM04FLTRE | channel 4 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01220$_H$ | GTM0TIM04FLTFE | channel 4 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01224$_H$ | GTM0TIM04CTRL | channel 4 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01228$_H$ | GTM0TIM04ECTRL | channel 4 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0122C$_H$ | GTM0TIM04IRQNOTIFY | channel 4 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01230$_H$ | GTM0TIM04IRQEN | channel 4 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01234$_H$ | GTM0TIM04IRQFORCINT | channel 4 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01238$_H$ | GTM0TIM04IRQMODE | IRQ mode configuration register 4 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0123C$_H$ | GTM0TIM04EIRQEN | channel 4 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01280$_H$ | GTM0TIM05GPR0 | channel 5 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01284$_H$ | GTM0TIM05GPR1 | channel 5 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01288$_H$ | GTM0TIM05CNT | channel 5 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (7/37)

| Module name | Address | Symbol | Register Name | Access Width | Support 207 | Support 208 | Access Protection PBG | Access Protection Other |
|---|---|---|---|---|---|---|---|---|
| GTM0 | <GTM0_base> + 0128C$_H$ | GTM0TIM05ECNT | channel 5 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01290$_H$ | GTM0TIM05CNTS | channel 5 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01294$_H$ | GTM0TIM05TDUC | channel 5 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01298$_H$ | GTM0TIM05TDUV | channel 5 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0129C$_H$ | GTM0TIM05FLTRE | channel 5 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 012A0$_H$ | GTM0TIM05FLTFE | channel 5 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 012A4$_H$ | GTM0TIM05CTRL | channel 5 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 012A8$_H$ | GTM0TIM05ECTRL | channel 5 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 012AC$_H$ | GTM0TIM05IRQNOTIFY | channel 5 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 012B0$_H$ | GTM0TIM05IRQEN | channel 5 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 012B4$_H$ | GTM0TIM05IRQFORCINT | channel 5 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 012B8$_H$ | GTM0TIM05IRQMODE | IRQ mode configuration register 5 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 012BC$_H$ | GTM0TIM05EIRQEN | channel 5 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01300$_H$ | GTM0TIM06GPR0 | channel 6 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01304$_H$ | GTM0TIM06GPR1 | channel 6 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01308$_H$ | GTM0TIM06CNT | channel 6 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0130C$_H$ | GTM0TIM06ECNT | channel 6 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01310$_H$ | GTM0TIM06CNTS | channel 6 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01314$_H$ | GTM0TIM06TDUC | channel 6 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01318$_H$ | GTM0TIM06TDUV | channel 6 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0131C$_H$ | GTM0TIM06FLTRE | channel 6 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01320$_H$ | GTM0TIM06FLTFE | channel 6 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01324$_H$ | GTM0TIM06CTRL | channel 6 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01328$_H$ | GTM0TIM06ECTRL | channel 6 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0132C$_H$ | GTM0TIM06IRQNOTIFY | channel 6 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01330$_H$ | GTM0TIM06IRQEN | channel 6 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01334$_H$ | GTM0TIM06IRQFORCINT | channel 6 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (8/37)

| Module name | Address | Symbol | Register Name | Access Width | Support 207 | Support 208 | Access Protection PBG | Access Protection Other |
|---|---|---|---|---|---|---|---|---|
| GTM0 | <GTM0_base> + 01338$_H$ | GTM0TIM06IRQMODE | IRQ mode configuration register 6 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0133C$_H$ | GTM0TIM06EIRQEN | channel 6 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01380$_H$ | GTM0TIM07GPR0 | channel 7 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01384$_H$ | GTM0TIM07GPR1 | channel 7 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01388$_H$ | GTM0TIM07CNT | channel 7 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0138C$_H$ | GTM0TIM07ECNT | channel 7 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01390$_H$ | GTM0TIM07CNTS | channel 7 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01394$_H$ | GTM0TIM07TDUC | channel 7 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01398$_H$ | GTM0TIM07TDUV | channel 7 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0139C$_H$ | GTM0TIM07FLTRE | channel 7 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 013A0$_H$ | GTM0TIM07FLTFE | channel 7 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 013A4$_H$ | GTM0TIM07CTRL | channel 7 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 013A8$_H$ | GTM0TIM07ECTRL | channel 7 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 013AC$_H$ | GTM0TIM07IRQNOTIFY | channel 7 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 013B0$_H$ | GTM0TIM07IRQEN | channel 7 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 013B4$_H$ | GTM0TIM07IRQFORCINT | channel 7 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 013B8$_H$ | GTM0TIM07IRQMODE | IRQ mode configuration register 7 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 013BC$_H$ | GTM0TIM07EIRQEN | channel 7 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01800$_H$ | GTM0TIM10GPR0 | channel 0 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01804$_H$ | GTM0TIM10GPR1 | channel 0 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01808$_H$ | GTM0TIM10CNT | channel 0 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0180C$_H$ | GTM0TIM10ECNT | channel 0 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01810$_H$ | GTM0TIM10CNTS | channel 0 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01814$_H$ | GTM0TIM10TDUC | channel 0 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01818$_H$ | GTM0TIM10TDUV | channel 0 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0181C$_H$ | GTM0TIM10FLTRE | channel 0 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01820$_H$ | GTM0TIM10FLTFE | channel 0 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (9/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 01824$_H$ | GTM0TIM10CTRL | channel 0 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01828$_H$ | GTM0TIM10ECTRL | channel 0 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0182C$_H$ | GTM0TIM10IRQNOTIFY | channel 0 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01830$_H$ | GTM0TIM10IRQEN | channel 0 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01834$_H$ | GTM0TIM10IRQFORCINT | channel 0 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01838$_H$ | GTM0TIM10IRQMODE | IRQ mode configuration register 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0183C$_H$ | GTM0TIM10EIRQEN | channel 0 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01874$_H$ | GTM0TIM1INPVAL | TIM input value observation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01878$_H$ | GTM0TIM1INSRC | TIM AUX IN source selection | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0187C$_H$ | GTM0TIM1RST | TIM global software reset | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01880$_H$ | GTM0TIM11GPR0 | channel 1 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01884$_H$ | GTM0TIM11GPR1 | channel 1 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01888$_H$ | GTM0TIM11CNT | channel 1 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0188C$_H$ | GTM0TIM11ECNT | channel 1 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01890$_H$ | GTM0TIM11CNTS | channel 1 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01894$_H$ | GTM0TIM11TDUC | channel 1 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01898$_H$ | GTM0TIM11TDUV | channel 1 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0189C$_H$ | GTM0TIM11FLTRE | channel 1 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 018A0$_H$ | GTM0TIM11FLTFE | channel 1 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 018A4$_H$ | GTM0TIM11CTRL | channel 1 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 018A8$_H$ | GTM0TIM11ECTRL | channel 1 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 018AC$_H$ | GTM0TIM11IRQNOTIFY | channel 1 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 018B0$_H$ | GTM0TIM11IRQEN | channel 1 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 018B4$_H$ | GTM0TIM11IRQFORCINT | channel 1 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 018B8$_H$ | GTM0TIM11IRQMODE | IRQ mode configuration register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 018BC$_H$ | GTM0TIM11EIRQEN | channel 1 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01900$_H$ | GTM0TIM12GPR0 | channel 2 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (10/37)

| Module name | Address | Symbol | Register Name | Access Width | Support 207 | Support 208 | Access Protection PBG | Access Protection Other |
|---|---|---|---|---|---|---|---|---|
| GTM0 | <GTM0_base> + $01904_H$ | GTM0TIM12GPR1 | channel 2 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01908_H$ | GTM0TIM12CNT | channel 2 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0190C_H$ | GTM0TIM12ECNT | channel 2 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01910_H$ | GTM0TIM12CNTS | channel 2 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01914_H$ | GTM0TIM12TDUC | channel 2 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01918_H$ | GTM0TIM12TDUV | channel 2 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0191C_H$ | GTM0TIM12FLTRE | channel 2 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01920_H$ | GTM0TIM12FLTFE | channel 2 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01924_H$ | GTM0TIM12CTRL | channel 2 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01928_H$ | GTM0TIM12ECTRL | channel 2 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0192C_H$ | GTM0TIM12IRQNOTIFY | channel 2 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01930_H$ | GTM0TIM12IRQEN | channel 2 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01934_H$ | GTM0TIM12IRQFORCINT | channel 2 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01938_H$ | GTM0TIM12IRQMODE | IRQ mode configuration register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0193C_H$ | GTM0TIM12EIRQEN | channel 2 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01980_H$ | GTM0TIM13GPR0 | channel 3 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01984_H$ | GTM0TIM13GPR1 | channel 3 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01988_H$ | GTM0TIM13CNT | channel 3 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0198C_H$ | GTM0TIM13ECNT | channel 3 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01990_H$ | GTM0TIM13CNTS | channel 3 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01994_H$ | GTM0TIM13TDUC | channel 3 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $01998_H$ | GTM0TIM13TDUV | channel 3 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $0199C_H$ | GTM0TIM13FLTRE | channel 3 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $019A0_H$ | GTM0TIM13FLTFE | channel 3 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $019A4_H$ | GTM0TIM13CTRL | channel 3 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $019A8_H$ | GTM0TIM13ECTRL | channel 3 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $019AC_H$ | GTM0TIM13IRQNOTIFY | channel 3 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (11/37)

| Module name | Address | Symbol | Register Name | Access Width | Support 207 | Support 208 | Access Protection PBG | Access Protection Other |
|---|---|---|---|---|---|---|---|---|
| GTM0 | <GTM0_base> + 019B0$_H$ | GTM0TIM13IRQEN | channel 3 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 019B4$_H$ | GTM0TIM13IRQFORCINT | channel 3 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 019B8$_H$ | GTM0TIM13IRQMODE | IRQ mode configuration register 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 019BC$_H$ | GTM0TIM13EIRQEN | channel 3 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A00$_H$ | GTM0TIM14GPR0 | channel 4 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A04$_H$ | GTM0TIM14GPR1 | channel 4 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A08$_H$ | GTM0TIM14CNT | channel 4 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A0C$_H$ | GTM0TIM14ECNT | channel 4 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A10$_H$ | GTM0TIM14CNTS | channel 4 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A14$_H$ | GTM0TIM14TDUC | channel 4 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A18$_H$ | GTM0TIM14TDUV | channel 4 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A1C$_H$ | GTM0TIM14FLTRE | channel 4 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A20$_H$ | GTM0TIM14FLTFE | channel 4 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A24$_H$ | GTM0TIM14CTRL | channel 4 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A28$_H$ | GTM0TIM14ECTRL | channel 4 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A2C$_H$ | GTM0TIM14IRQNOTIFY | channel 4 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A30$_H$ | GTM0TIM14IRQEN | channel 4 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A34$_H$ | GTM0TIM14IRQFORCINT | channel 4 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A38$_H$ | GTM0TIM14IRQMODE | IRQ mode configuration register 4 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A3C$_H$ | GTM0TIM14EIRQEN | channel 4 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A80$_H$ | GTM0TIM15GPR0 | channel 5 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A84$_H$ | GTM0TIM15GPR1 | channel 5 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A88$_H$ | GTM0TIM15CNT | channel 5 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A8C$_H$ | GTM0TIM15ECNT | channel 5 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A90$_H$ | GTM0TIM15CNTS | channel 5 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A94$_H$ | GTM0TIM15TDUC | channel 5 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01A98$_H$ | GTM0TIM15TDUV | channel 5 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (12/37)

| Module name | Address | Symbol | Register Name | Access Width | Support 207 | Support 208 | Access Protection PBG | Access Protection Other |
|---|---|---|---|---|---|---|---|---|
| GTM0 | <GTM0_base> + 01A9C$_H$ | GTM0TIM15FLTRE | channel 5 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01AA0$_H$ | GTM0TIM15FLTFE | channel 5 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01AA4$_H$ | GTM0TIM15CTRL | channel 5 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01AA8$_H$ | GTM0TIM15ECTRL | channel 5 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01AAC$_H$ | GTM0TIM15IRQNOTIFY | channel 5 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01AB0$_H$ | GTM0TIM15IRQEN | channel 5 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01AB4$_H$ | GTM0TIM15IRQFORCINT | channel 5 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01AB8$_H$ | GTM0TIM15IRQMODE | IRQ mode configuration register 5 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01ABC$_H$ | GTM0TIM15EIRQEN | channel 5 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B00$_H$ | GTM0TIM16GPR0 | channel 6 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B04$_H$ | GTM0TIM16GPR1 | channel 6 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B08$_H$ | GTM0TIM16CNT | channel 6 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B0C$_H$ | GTM0TIM16ECNT | channel 6 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B10$_H$ | GTM0TIM16CNTS | channel 6 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B14$_H$ | GTM0TIM16TDUC | channel 6 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B18$_H$ | GTM0TIM16TDUV | channel 6 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B1C$_H$ | GTM0TIM16FLTRE | channel 6 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B20$_H$ | GTM0TIM16FLTFE | channel 6 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B24$_H$ | GTM0TIM16CTRL | channel 6 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B28$_H$ | GTM0TIM16ECTRL | channel 6 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B2C$_H$ | GTM0TIM16IRQNOTIFY | channel 6 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B30$_H$ | GTM0TIM16IRQEN | channel 6 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B34$_H$ | GTM0TIM16IRQFORCINT | channel 6 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B38$_H$ | GTM0TIM16IRQMODE | IRQ mode configuration register 6 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B3C$_H$ | GTM0TIM16EIRQEN | channel 6 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B80$_H$ | GTM0TIM17GPR0 | channel 7 general purpose 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B84$_H$ | GTM0TIM17GPR1 | channel 7 general purpose 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (13/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 01B88$_H$ | GTM0TIM17CNT | channel 7 SMU counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B8C$_H$ | GTM0TIM17ECNT | channel 7 SMU edge counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B90$_H$ | GTM0TIM17CNTS | channel 7 SMU shadow counter | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B94$_H$ | GTM0TIM17TDUC | channel 7 TDU counter. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B98$_H$ | GTM0TIM17TDUV | channel 7 TDU control. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01B9C$_H$ | GTM0TIM17FLTRE | channel 7 filter parameter 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01BA0$_H$ | GTM0TIM17FLTFE | channel 7 filter parameter 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01BA4$_H$ | GTM0TIM17CTRL | channel 7 control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01BA8$_H$ | GTM0TIM17ECTRL | channel 7 extended control | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01BAC$_H$ | GTM0TIM17IRQNOTIFY | channel 7 interrupt notification | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01BB0$_H$ | GTM0TIM17IRQEN | channel 7 interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01BB4$_H$ | GTM0TIM17IRQFORCINT | channel 7 software interrupt force | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01BB8$_H$ | GTM0TIM17IRQMODE | IRQ mode configuration register 7 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 01BBC$_H$ | GTM0TIM17EIRQEN | channel 7 error interrupt enable | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D000$_H$ | GTM0ATOM00RDADDR | ATOM Channel 0 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D004$_H$ | GTM0ATOM00CTRL | ATOM Channel 0 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D008$_H$ | GTM0ATOM00SR0 | ATOM Channel 0 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D00C$_H$ | GTM0ATOM00SR1 | ATOM Channel 0 CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D010$_H$ | GTM0ATOM00CM0 | ATOM Channel 0 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D014$_H$ | GTM0ATOM00CM1 | ATOM Channel 0 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D018$_H$ | GTM0ATOM00CN0 | ATOM Channel 0 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D01C$_H$ | GTM0ATOM00STAT | ATOM Channel 0 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D020$_H$ | GTM0ATOM00IRQNOTIFY | ATOM channel 0 interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D024$_H$ | GTM0ATOM00IRQEN | ATOM channel 0 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D028$_H$ | GTM0ATOM00IRQFORCINT | ATOM channel 0 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D02C$_H$ | GTM0ATOM00IRQMODE | IRQ mode configuration register 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D040$_H$ | GTM0ATOM0AGCGLBCTRL | AGC Global control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (14/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 0D044$_H$ | GTM0ATOM0AGCENDISCTRL | AGC0 Enable/disable control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D048$_H$ | GTM0ATOM0AGCENDISSTAT | AGC Enable/disable status register (represents status of ATOM channels) | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D04C$_H$ | GTM0ATOM0AGCACTTB | AGC Action time base register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D050$_H$ | GTM0ATOM0AGCOUTENCTRL | AGC Output enable control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D054$_H$ | GTM0ATOM0AGCOUTENSTAT | AGC Output enable status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D058$_H$ | GTM0ATOM0AGCFUPDCTRL | AGC Force update control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D05C$_H$ | GTM0ATOM0AGCINTTRIG | AGC Internal trigger control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D080$_H$ | GTM0ATOM01RDADDR | ATOM Channel 1 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D084$_H$ | GTM0ATOM01CTRL | ATOM Channel 1 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D088$_H$ | GTM0ATOM01SR0 | ATOM Channel 1 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D08C$_H$ | GTM0ATOM01SR1 | ATOM Channel 1CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D090$_H$ | GTM0ATOM01CM0 | ATOM Channel 1 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D094$_H$ | GTM0ATOM01CM1 | ATOM Channel 1 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D098$_H$ | GTM0ATOM01CN0 | ATOM Channel 1 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D09C$_H$ | GTM0ATOM01STAT | ATOM Channel 1 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D0A0$_H$ | GTM0ATOM01IRQNOTIFY | ATOM channel 1 interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D0A4$_H$ | GTM0ATOM01IRQEN | ATOM channel 1 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D0A8$_H$ | GTM0ATOM01IRQFORCINT | ATOM channel 1 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D0AC$_H$ | GTM0ATOM01IRQMODE | IRQ mode configuration register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D100$_H$ | GTM0ATOM02RDADDR | ATOM Channel 2 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D104$_H$ | GTM0ATOM02CTRL | ATOM Channel 2 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D108$_H$ | GTM0ATOM02SR0 | ATOM Channel 2 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D10C$_H$ | GTM0ATOM02SR1 | ATOM Channel 2 CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D110$_H$ | GTM0ATOM02CM0 | ATOM Channel 2 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D114$_H$ | GTM0ATOM02CM1 | ATOM Channel 2 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D118$_H$ | GTM0ATOM02CN0 | ATOM Channel 2 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D11C$_H$ | GTM0ATOM02STAT | ATOM Channel 2 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (15/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 0D120$_H$ | GTM0ATOM02IRQNOTIFY | ATOM channel 2 interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D124$_H$ | GTM0ATOM02IRQEN | ATOM channel 2 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D128$_H$ | GTM0ATOM02IRQFORCINT | ATOM channel 2 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D12C$_H$ | GTM0ATOM02IRQMODE | IRQ mode configuration register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D180$_H$ | GTM0ATOM03RDADDR | ATOM Channel 3 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D184$_H$ | GTM0ATOM03CTRL | ATOM Channel 3 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D188$_H$ | GTM0ATOM03SR0 | ATOM Channel 3 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D18C$_H$ | GTM0ATOM03SR1 | ATOM Channel 3 CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D190$_H$ | GTM0ATOM03CM0 | ATOM Channel 3 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D194$_H$ | GTM0ATOM03CM1 | ATOM Channel 3 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D198$_H$ | GTM0ATOM03CN0 | ATOM Channel 3 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D19C$_H$ | GTM0ATOM03STAT | ATOM Channel 3 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D1A0$_H$ | GTM0ATOM03IRQNOTIFY | ATOM channel 3x interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D1A4$_H$ | GTM0ATOM03IRQEN | ATOM channel 3 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D1A8$_H$ | GTM0ATOM03IRQFORCINT | ATOM channel 3 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D1AC$_H$ | GTM0ATOM03IRQMODE | IRQ mode configuration register 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D200$_H$ | GTM0ATOM04RDADDR | ATOM Channel 4 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D204$_H$ | GTM0ATOM04CTRL | ATOM Channel 4 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D208$_H$ | GTM0ATOM04SR0 | ATOM Channel 4 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D20C$_H$ | GTM0ATOM04SR1 | ATOM Channel 4 CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D210$_H$ | GTM0ATOM04CM0 | ATOM Channel 4 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D214$_H$ | GTM0ATOM04CM1 | ATOM Channel 4 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D218$_H$ | GTM0ATOM04CN0 | ATOM Channel 4 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D21C$_H$ | GTM0ATOM04STAT | ATOM Channel 4 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D220$_H$ | GTM0ATOM04IRQNOTIFY | ATOM channel 4 interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D224$_H$ | GTM0ATOM04IRQEN | ATOM channel 4 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D228$_H$ | GTM0ATOM04IRQFORCINT | ATOM channel 4 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (16/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 0D22C$_H$ | GTM0ATOM04IRQMODE | IRQ mode configuration register 4 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D280$_H$ | GTM0ATOM05RDADDR | ATOM Channel 5 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D284$_H$ | GTM0ATOM05CTRL | ATOM Channel 5 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D288$_H$ | GTM0ATOM05SR0 | ATOM Channel 5 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D28C$_H$ | GTM0ATOM05SR1 | ATOM Channel 5 CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D290$_H$ | GTM0ATOM05CM0 | ATOM Channel 5 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D294$_H$ | GTM0ATOM05CM1 | ATOM Channel 5 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D298$_H$ | GTM0ATOM05CN0 | ATOM Channel 5 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D29C$_H$ | GTM0ATOM05STAT | ATOM Channel 5 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D2A0$_H$ | GTM0ATOM05IRQNOTIFY | ATOM channel 5 interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D2A4$_H$ | GTM0ATOM05IRQEN | ATOM channel 5 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D2A8$_H$ | GTM0ATOM05IRQFORCINT | ATOM channel 5 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D2AC$_H$ | GTM0ATOM05IRQMODE | IRQ mode configuration register 5 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D300$_H$ | GTM0ATOM06RDADDR | ATOM Channel 6 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D304$_H$ | GTM0ATOM06CTRL | ATOM Channel 6 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D308$_H$ | GTM0ATOM06SR0 | ATOM Channel 6 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D30C$_H$ | GTM0ATOM06SR1 | ATOM Channel 6 CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D310$_H$ | GTM0ATOM06CM0 | ATOM Channel 6 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D314$_H$ | GTM0ATOM06CM1 | ATOM Channel 6 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D318$_H$ | GTM0ATOM06CN0 | ATOM Channel 6 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D31C$_H$ | GTM0ATOM06STAT | ATOM Channel 6 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D320$_H$ | GTM0ATOM06IRQNOTIFY | ATOM channel 6 interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D324$_H$ | GTM0ATOM06IRQEN | ATOM channel 6 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D328$_H$ | GTM0ATOM06IRQFORCINT | ATOM channel 6 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D32C$_H$ | GTM0ATOM06IRQMODE | IRQ mode configuration register 6 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D380$_H$ | GTM0ATOM07RDADDR | ATOM Channel 7 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D384$_H$ | GTM0ATOM07CTRL | ATOM Channel 7 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (17/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 0D388$_H$ | GTM0ATOM07SR0 | ATOM Channel 7 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D38C$_H$ | GTM0ATOM07SR1 | ATOM Channel 7 CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D390$_H$ | GTM0ATOM07CM0 | ATOM Channel 7 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D394$_H$ | GTM0ATOM07CM1 | ATOM Channel 7 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D398$_H$ | GTM0ATOM07CN0 | ATOM Channel 7 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D39C$_H$ | GTM0ATOM07STAT | ATOM Channel 7 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D3A0$_H$ | GTM0ATOM07IRQNOTIFY | ATOM channel 7 interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D3A4$_H$ | GTM0ATOM07IRQEN | ATOM channel 7 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D3A8$_H$ | GTM0ATOM07IRQFORCINT | ATOM channel 7 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D3AC$_H$ | GTM0ATOM07IRQMODE | IRQ mode configuration register 7 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D800$_H$ | GTM0ATOM10RDADDR | ATOM Channel 0 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D804$_H$ | GTM0ATOM10CTRL | ATOM Channel 0 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D808$_H$ | GTM0ATOM10SR0 | ATOM Channel 0 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D80C$_H$ | GTM0ATOM10SR1 | ATOM Channel 0 CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D810$_H$ | GTM0ATOM10CM0 | ATOM Channel 0 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D814$_H$ | GTM0ATOM10CM1 | ATOM Channel 0 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D818$_H$ | GTM0ATOM10CN0 | ATOM Channel 0 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D81C$_H$ | GTM0ATOM10STAT | ATOM Channel 0 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D820$_H$ | GTM0ATOM10IRQNOTIFY | ATOM channel 0 interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D824$_H$ | GTM0ATOM10IRQEN | ATOM channel 0 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D828$_H$ | GTM0ATOM10IRQFORCINT | ATOM channel 0 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D82C$_H$ | GTM0ATOM10IRQMODE | IRQ mode configuration register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D840$_H$ | GTM0ATOM1AGCGLBCTRL | AGC Global control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D844$_H$ | GTM0ATOM1AGCENDISCTRL | AGC0 Enable/disable control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D848$_H$ | GTM0ATOM1AGCENDISSTAT | AGC Enable/disable status register (represents status of ATOM channels) | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D84C$_H$ | GTM0ATOM1AGCACTTB | AGC Action time base register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D850$_H$ | GTM0ATOM1AGCOUTENCTRL | AGC Output enable control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (18/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 0D854$_H$ | GTM0ATOM1AGCOUTENSTAT | AGC Output enable status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D858$_H$ | GTM0ATOM1AGCFUPDCTRL | AGC Force update control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D85C$_H$ | GTM0ATOM1AGCINTTRIG | AGC Internal trigger control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D880$_H$ | GTM0ATOM11RDADDR | ATOM Channel 1 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D884$_H$ | GTM0ATOM11CTRL | ATOM Channel 1 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D888$_H$ | GTM0ATOM11SR0 | ATOM Channel 1 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D88C$_H$ | GTM0ATOM11SR1 | ATOM Channel 1 CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D890$_H$ | GTM0ATOM11CM0 | ATOM Channel 1 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D894$_H$ | GTM0ATOM11CM1 | ATOM Channel 1 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D898$_H$ | GTM0ATOM11CN0 | ATOM Channel 1 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D89C$_H$ | GTM0ATOM11STAT | ATOM Channel 1 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D8A0$_H$ | GTM0ATOM11IRQNOTIFY | ATOM channel 1 interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D8A4$_H$ | GTM0ATOM11IRQEN | ATOM channel 1 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D8A8$_H$ | GTM0ATOM11IRQFORCINT | ATOM channel 1 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D8AC$_H$ | GTM0ATOM11IRQMODE | IRQ mode configuration register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D900$_H$ | GTM0ATOM12RDADDR | ATOM Channel 2 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D904$_H$ | GTM0ATOM12CTRL | ATOM Channel 2 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D908$_H$ | GTM0ATOM12SR0 | ATOM Channel 2 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D90C$_H$ | GTM0ATOM12SR1 | ATOM Channel 2 CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D910$_H$ | GTM0ATOM12CM0 | ATOM Channel 2 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D914$_H$ | GTM0ATOM12CM1 | ATOM Channel 2 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D918$_H$ | GTM0ATOM12CN0 | ATOM Channel 2 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D91C$_H$ | GTM0ATOM12STAT | ATOM Channel 2 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D920$_H$ | GTM0ATOM12IRQNOTIFY | ATOM channel 2 interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D924$_H$ | GTM0ATOM12IRQEN | ATOM channel 2 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D928$_H$ | GTM0ATOM12IRQFORCINT | ATOM channel 2 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D92C$_H$ | GTM0ATOM12IRQMODE | IRQ mode configuration register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (19/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 0D980$_H$ | GTM0ATOM13RDADDR | ATOM Channel 3 ARU read address register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D984$_H$ | GTM0ATOM13CTRL | ATOM Channel 3 control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D988$_H$ | GTM0ATOM13SR0 | ATOM Channel 3 CCU0 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D98C$_H$ | GTM0ATOM13SR1 | ATOM Channel 3 CCU1 compare shadow register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D990$_H$ | GTM0ATOM13CM0 | ATOM Channel 3 CCU0 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D994$_H$ | GTM0ATOM13CM1 | ATOM Channel 3 CCU1 compare register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D998$_H$ | GTM0ATOM13CN0 | ATOM Channel 3 CCU0 counter register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D99C$_H$ | GTM0ATOM13STAT | ATOM Channel 3 status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D9A0$_H$ | GTM0ATOM13IRQNOTIFY | ATOM channel 3 interrupt notification register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D9A4$_H$ | GTM0ATOM13IRQEN | ATOM channel 3 interrupt enable register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D9A8$_H$ | GTM0ATOM13IRQFORCINT | ATOM channel 3 software interrupt generation | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0D9AC$_H$ | GTM0ATOM13IRQMODE | IRQ mode configuration register 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA00$_H$ | GTM0ATOM14RDADDR | ATOM Channel 4 ARU read address register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA04$_H$ | GTM0ATOM14CTRL | ATOM Channel 4 control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA08$_H$ | GTM0ATOM14SR0 | ATOM Channel 4 CCU0 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA0C$_H$ | GTM0ATOM14SR1 | ATOM Channel 4 CCU1 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA10$_H$ | GTM0ATOM14CM0 | ATOM Channel 4 CCU0 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA14$_H$ | GTM0ATOM14CM1 | ATOM Channel 4 CCU1 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA18$_H$ | GTM0ATOM14CN0 | ATOM Channel 4 CCU0 counter register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA1C$_H$ | GTM0ATOM14STAT | ATOM Channel 4 status register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA20$_H$ | GTM0ATOM14IRQNOTIFY | ATOM channel 4 interrupt notification register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA24$_H$ | GTM0ATOM14IRQEN | ATOM channel 4 interrupt enable register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA28$_H$ | GTM0ATOM14IRQFORCINT | ATOM channel 4 software interrupt generation | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA2C$_H$ | GTM0ATOM14IRQMODE | IRQ mode configuration register 4 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA80$_H$ | GTM0ATOM15RDADDR | ATOM Channel 5 ARU read address register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA84$_H$ | GTM0ATOM15CTRL | ATOM Channel 5 control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA88$_H$ | GTM0ATOM15SR0 | ATOM Channel 5 CCU0 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (20/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 0DA8C$_H$ | GTM0ATOM15SR1 | ATOM Channel 5 CCU1 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA90$_H$ | GTM0ATOM15CM0 | ATOM Channel 5 CCU0 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA94$_H$ | GTM0ATOM15CM1 | ATOM Channel 5 CCU1 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA98$_H$ | GTM0ATOM15CN0 | ATOM Channel 5 CCU0 counter register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DA9C$_H$ | GTM0ATOM15STAT | ATOM Channel 5 status register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DAA0$_H$ | GTM0ATOM15IRQNOTIFY | ATOM channel 5 interrupt notification register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DAA4$_H$ | GTM0ATOM15IRQEN | ATOM channel 5 interrupt enable register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DAA8$_H$ | GTM0ATOM15IRQFORCINT | ATOM channel 5 software interrupt generation | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DAAC$_H$ | GTM0ATOM15IRQMODE | IRQ mode configuration register 5 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB00$_H$ | GTM0ATOM16RDADDR | ATOM Channel 6 ARU read address register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB04$_H$ | GTM0ATOM16CTRL | ATOM Channel 6 control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB08$_H$ | GTM0ATOM16SR0 | ATOM Channel 6 CCU0 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB0C$_H$ | GTM0ATOM16SR1 | ATOM Channel 6 CCU1 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB10$_H$ | GTM0ATOM16CM0 | ATOM Channel 6 CCU0 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB14$_H$ | GTM0ATOM16CM1 | ATOM Channel 6 CCU1 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB18$_H$ | GTM0ATOM16CN0 | ATOM Channel 6 CCU0 counter register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB1C$_H$ | GTM0ATOM16STAT | ATOM Channel 6 status register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB20$_H$ | GTM0ATOM16IRQNOTIFY | ATOM channel 6 interrupt notification register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB24$_H$ | GTM0ATOM16IRQEN | ATOM channel 6 interrupt enable register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB28$_H$ | GTM0ATOM16IRQFORCINT | ATOM channel 6 software interrupt generation | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB2C$_H$ | GTM0ATOM16IRQMODE | IRQ mode configuration register 6 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB80$_H$ | GTM0ATOM17RDADDR | ATOM Channel 7 ARU read address register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB84$_H$ | GTM0ATOM17CTRL | ATOM Channel 7 control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB88$_H$ | GTM0ATOM17SR0 | ATOM Channel 7 CCU0 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB8C$_H$ | GTM0ATOM17SR1 | ATOM Channel 7 CCU1 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB90$_H$ | GTM0ATOM17CM0 | ATOM Channel 7 CCU0 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB94$_H$ | GTM0ATOM17CM1 | ATOM Channel 7 CCU1 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (21/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 0DB98$_H$ | GTM0ATOM17CN0 | ATOM Channel 7 CCU0 counter register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DB9C$_H$ | GTM0ATOM17STAT | ATOM Channel 7 status register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DBA0$_H$ | GTM0ATOM17IRQNOTIFY | ATOM channel 7 interrupt notification register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DBA4$_H$ | GTM0ATOM17IRQEN | ATOM channel 7 interrupt enable register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DBA8$_H$ | GTM0ATOM17IRQFORCINT | ATOM channel 7 software interrupt generation | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0DBAC$_H$ | GTM0ATOM17IRQMODE | IRQ mode configuration register 7 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E000$_H$ | GTM0ATOM20RDADDR | ATOM Channel 0 ARU read address register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E004$_H$ | GTM0ATOM20CTRL | ATOM Channel 0 control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E008$_H$ | GTM0ATOM20SR0 | ATOM Channel 0 CCU0 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E00C$_H$ | GTM0ATOM20SR1 | ATOM Channel 0 CCU1 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E010$_H$ | GTM0ATOM20CM0 | ATOM Channel 0 CCU0 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E014$_H$ | GTM0ATOM20CM1 | ATOM Channel 0 CCU1 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E018$_H$ | GTM0ATOM20CN0 | ATOM Channel 0 CCU0 counter register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E01C$_H$ | GTM0ATOM20STAT | ATOM Channel 0 status register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E020$_H$ | GTM0ATOM20IRQNOTIFY | ATOM channel 0 interrupt notification register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E024$_H$ | GTM0ATOM20IRQEN | ATOM channel 0 interrupt enable register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E028$_H$ | GTM0ATOM20IRQFORCINT | ATOM channel 0 software interrupt generation | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E02C$_H$ | GTM0ATOM20IRQMODE | IRQ mode configuration register 0 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E040$_H$ | GTM0ATOM2AGCGLBCTRL | AGC Global control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E044$_H$ | GTM0ATOM2AGCENDISCTRL | AGC0 Enable/disable control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E048$_H$ | GTM0ATOM2AGCENDISSTAT | AGC Enable/disable status register (represents status of ATOM channels) | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E04C$_H$ | GTM0ATOM2AGCACTTB | AGC Action time base register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E050$_H$ | GTM0ATOM2AGCOUTENCTRL | AGC Output enable control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E054$_H$ | GTM0ATOM2AGCOUTENSTAT | AGC Output enable status register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E058$_H$ | GTM0ATOM2AGCFUPDCTRL | AGC Force update control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E05C$_H$ | GTM0ATOM2AGCINTTRIG | AGC Internal trigger control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E080$_H$ | GTM0ATOM21RDADDR | ATOM Channel 1 ARU read address register | 32 | — | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (22/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 0E084$_H$ | GTM0ATOM21CTRL | ATOM Channel 1 control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E088$_H$ | GTM0ATOM21SR0 | ATOM Channel 1 CCU0 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E08C$_H$ | GTM0ATOM21SR1 | ATOM Channel 1 CCU1 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E090$_H$ | GTM0ATOM21CM0 | ATOM Channel 1 CCU0 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E094$_H$ | GTM0ATOM21CM1 | ATOM Channel 1 CCU1 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E098$_H$ | GTM0ATOM21CN0 | ATOM Channel 1 CCU0 counter register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E09C$_H$ | GTM0ATOM21STAT | ATOM Channel 1 status register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E0A0$_H$ | GTM0ATOM21IRQNOTIFY | ATOM channel 1 interrupt notification register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E0A4$_H$ | GTM0ATOM21IRQEN | ATOM channel 1 interrupt enable register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E0A8$_H$ | GTM0ATOM21IRQFORCINT | ATOM channel 1 software interrupt generation | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E0AC$_H$ | GTM0ATOM21IRQMODE | IRQ mode configuration register 1 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E100$_H$ | GTM0ATOM22RDADDR | ATOM Channel 2 ARU read address register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E104$_H$ | GTM0ATOM22CTRL | ATOM Channel 2 control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E108$_H$ | GTM0ATOM22SR0 | ATOM Channel 2 CCU0 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E10C$_H$ | GTM0ATOM22SR1 | ATOM Channel 2 CCU1 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E110$_H$ | GTM0ATOM22CM0 | ATOM Channel 2 CCU0 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E114$_H$ | GTM0ATOM22CM1 | ATOM Channel 2 CCU1 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E118$_H$ | GTM0ATOM22CN0 | ATOM Channel 2 CCU0 counter register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E11C$_H$ | GTM0ATOM22STAT | ATOM Channel 2 status register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E120$_H$ | GTM0ATOM22IRQNOTIFY | ATOM channel 2 interrupt notification register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E124 | GTM0ATOM22IRQEN | ATOM channel 2 interrupt enable register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E128$_H$ | GTM0ATOM22IRQFORCINT | ATOM channel 2 software interrupt generation | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E12C$_H$ | GTM0ATOM22IRQMODE | IRQ mode configuration register 2 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E180$_H$ | GTM0ATOM23RDADDR | ATOM Channel 3 ARU read address register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E184$_H$ | GTM0ATOM23CTRL | ATOM Channel 3 control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E188$_H$ | GTM0ATOM23SR0 | ATOM Channel 3 CCU0 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E18C$_H$ | GTM0ATOM23SR1 | ATOM Channel 3 CCU1 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (23/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 0E190$_H$ | GTM0ATOM23CM0 | ATOM Channel 3 CCU0 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E194$_H$ | GTM0ATOM23CM1 | ATOM Channel 3 CCU1 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E198$_H$ | GTM0ATOM23CN0 | ATOM Channel 3 CCU0 counter register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E19C$_H$ | GTM0ATOM23STAT | ATOM Channel 3 status register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E1A0$_H$ | GTM0ATOM23IRQNOTIFY | ATOM channel 3 interrupt notification register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E1A4$_H$ | GTM0ATOM23IRQEN | ATOM channel 3 interrupt enable register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E1A8$_H$ | GTM0ATOM23IRQFORCINT | ATOM channel 3 software interrupt generation | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E1AC$_H$ | GTM0ATOM23IRQMODE | IRQ mode configuration register 3 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E200$_H$ | GTM0ATOM24RDADDR | ATOM Channel 4 ARU read address register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E204$_H$ | GTM0ATOM24CTRL | ATOM Channel 4 control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E208$_H$ | GTM0ATOM24SR0 | ATOM Channel 4 CCU0 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E20C$_H$ | GTM0ATOM24SR1 | ATOM Channel 4 CCU1 compare shadow register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E210$_H$ | GTM0ATOM24CM0 | ATOM Channel 4 CCU0 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E214$_H$ | GTM0ATOM24CM1 | ATOM Channel 4 CCU1 compare register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E218$_H$ | GTM0ATOM24CN0 | ATOM Channel 4 CCU0 counter register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E21C$_H$ | GTM0ATOM24STAT | ATOM Channel 4 status register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E220$_H$ | GTM0ATOM24IRQNOTIFY | ATOM channel 4 interrupt notification register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E224$_H$ | GTM0ATOM24IRQEN | ATOM channel 4 interrupt enable register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E228$_H$ | GTM0ATOM24IRQFORCINT | ATOM channel 4 software interrupt generation | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 0E22C$_H$ | GTM0ATOM24IRQMODE | IRQ mode configuration register 4 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13600$_H$ | GTM0DTM24CTRL | Global Configuration and Control Register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13604$_H$ | GTM0DTM24CHCTRL1 | Channel Control Register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13608$_H$ | GTM0DTM24CHCTRL2 | Channel Control Register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 1360C$_H$ | GTM0DTM24CHCTRL2SR | Channel Control Register 2 Shadow | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13610$_H$ | GTM0DTM24PSCTRL | Phase Shift Unit Configuration and Control Register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13614$_H$ | GTM0DTM240DTV | Dead Time Reload Values 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13618$_H$ | GTM0DTM241DTV | Dead Time Reload Values 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (24/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 1361C$_H$ | GTM0DTM242DTV | Dead Time Reload Values 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13620$_H$ | GTM0DTM243DTV | Dead Time Reload Values 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13624$_H$ | GTM0DTM24CHSR | Channel Shadow Register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13680$_H$ | GTM0DTM26CTRL | Global Configuration and Control Register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13684$_H$ | GTM0DTM26CHCTRL1 | Channel Control Register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13688$_H$ | GTM0DTM26CHCTRL2 | Channel Control Register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 1368C$_H$ | GTM0DTM26CHCTRL2SR | Channel Control Register 2 Shadow | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13690$_H$ | GTM0DTM26PSCTRL | Phase Shift Unit Configuration and Control Register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13694$_H$ | GTM0DTM260DTV | Dead Time Reload Values 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13698$_H$ | GTM0DTM261DTV | Dead Time Reload Values 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 1369C$_H$ | GTM0DTM262DTV | Dead Time Reload Values 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 136A0$_H$ | GTM0DTM263DTV | Dead Time Reload Values 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 136A4$_H$ | GTM0DTM26CHSR | Channel Shadow Register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13700$_H$ | GTM0DTM28CTRL | Global Configuration and Control Register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13704$_H$ | GTM0DTM28CHCTRL1 | Channel Control Register 1 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13708$_H$ | GTM0DTM28CHCTRL2 | Channel Control Register 2 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 1370C$_H$ | GTM0DTM28CHCTRL2SR | Channel Control Register 2 Shadow | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13710$_H$ | GTM0DTM28PSCTRL | Phase Shift Unit Configuration and Control Register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13714$_H$ | GTM0DTM280DTV | Dead Time Reload Values 0 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13718$_H$ | GTM0DTM281DTV | Dead Time Reload Values 1 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 1371C$_H$ | GTM0DTM282DTV | Dead Time Reload Values 2 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13720$_H$ | GTM0DTM283DTV | Dead Time Reload Values 3 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 13724$_H$ | GTM0DTM28CHSR | Channel Shadow Register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30000$_H$ | GTM0MCS00R0 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R0. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30004$_H$ | GTM0MCS00R1 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R1. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30008$_H$ | GTM0MCS00R2 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R2. | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (25/37)

| Module name | Address | Symbol | Register Name | Access Width | Support 207 | Support 208 | Access Protection PBG | Access Protection Other |
|---|---|---|---|---|---|---|---|---|
| GTM0 | <GTM0_base> + 3000C$_H$ | GTM0MCS00R3 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R3. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30010$_H$ | GTM0MCS00R4 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R4. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30014$_H$ | GTM0MCS00R5 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R5. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30018$_H$ | GTM0MCS00R6 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R6. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3001C$_H$ | GTM0MCS00R7 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R7. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30020$_H$ | GTM0MCS00CTRL | MCS Channel control register 0. Most bits mirror the internal task specific register STA. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30024$_H$ | GTM0MCS00ACB | MCS Channel ACB register 0. The Register mirrors the internal task specific register ACB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30028$_H$ | GTM0MCS0CTRG | MCS Clear trigger control register. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3002C$_H$ | GTM0MCS0STRG | MCS Set trigger control register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3003C$_H$ | GTM0MCS00MHB | Memory High Byte register 0. The Register mirrors the internal task specific register MHB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30040$_H$ | GTM0MCS00PC | MCS Channel Program counter register 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30044$_H$ | GTM0MCS00IRQNOTIFY | MCS Channel x interrupt notification register 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30048$_H$ | GTM0MCS00IRQEN | MCS Channel x interrupt enable register 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3004C$_H$ | GTM0MCS00IRQFORCINT | MCS Channel x software interrupt generation register 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30050$_H$ | GTM0MCS00IRQMODE | IRQ mode configuration register 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30054$_H$ | GTM0MCS00EIRQEN | MCS Channel x error interrupt enable register 0 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30064$_H$ | GTM0MCS0CTRLSTAT | MCS Control and Status register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30068$_H$ | GTM0MCS0RESET | MCS Channel reset register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3006C$_H$ | GTM0MCS0CAT | Cancel ARU transfer register. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30070$_H$ | GTM0MCS0CWT | Cancel WURM instruction. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3007C$_H$ | GTM0MCS0ERR | MCS Error register | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30080$_H$ | GTM0MCS01R0 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R0. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30084$_H$ | GTM0MCS01R1 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R1. | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (26/37)

| Module name | Address | Symbol | Register Name | Access Width | Support 207 | Support 208 | Access Protection PBG | Access Protection Other |
|---|---|---|---|---|---|---|---|---|
| GTM0 | <GTM0_base> + $30088_H$ | GTM0MCS01R2 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R2. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3008C_H$ | GTM0MCS01R3 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R3. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30090_H$ | GTM0MCS01R4 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R4. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30094_H$ | GTM0MCS01R5 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R5. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30098_H$ | GTM0MCS01R6 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R6. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3009C_H$ | GTM0MCS01R7 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R7. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $300A0_H$ | GTM0MCS01CTRL | MCS Channel control register 1. Most bits mirror the internal task specific register STA. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $300A4_H$ | GTM0MCS01ACB | MCS Channel ACB register 1. The Register mirrors the internal task specific register ACB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $300BC_H$ | GTM0MCS01MHB | Memory High Byte register 1. The Register mirrors the internal task specific register MHB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $300C0_H$ | GTM0MCS01PC | MCS Channel Program counter register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $300C4_H$ | GTM0MCS01IRQNOTIFY | MCS Channel x interrupt notification register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $300C8_H$ | GTM0MCS01IRQEN | MCS Channel x interrupt enable register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $300CC_H$ | GTM0MCS01IRQFORCINT | MCS Channel x software interrupt generation register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $300D0_H$ | GTM0MCS01IRQMODE | IRQ mode configuration register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $300D4_H$ | GTM0MCS01EIRQEN | MCS Channel x error interrupt enable register 1 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30100_H$ | GTM0MCS02R0 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R0. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30104_H$ | GTM0MCS02R1 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R1. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30108_H$ | GTM0MCS02R2 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R2. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3010C_H$ | GTM0MCS02R3 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R3. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30110_H$ | GTM0MCS02R4 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R4. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30114_H$ | GTM0MCS02R5 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R5. | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (27/37)

| Module name | Address | Symbol | Register Name | Access Width | Support 207 | Support 208 | Access Protection PBG | Access Protection Other |
|---|---|---|---|---|---|---|---|---|
| GTM0 | <GTM0_base> + $30118_H$ | GTM0MCS02R6 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R6. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3011C_H$ | GTM0MCS02R7 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R7. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30120_H$ | GTM0MCS02CTRL | MCS Channel control register 2. Most bits mirror the internal task specific register STA. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30124_H$ | GTM0MCS02ACB | MCS Channel ACB register 2. The Register mirrors the internal task specific register ACB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3013C_H$ | GTM0MCS02MHB | Memory High Byte register 2. The Register mirrors the internal task specific register MHB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30140_H$ | GTM0MCS02PC | MCS Channel Program counter register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30144_H$ | GTM0MCS02IRQNOTIFY | MCS Channel x interrupt notification register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30148_H$ | GTM0MCS02IRQEN | MCS Channel x interrupt enable register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3014C_H$ | GTM0MCS02IRQFORCINT | MCS Channel x software interrupt generation register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30150_H$ | GTM0MCS02IRQMODE | IRQ mode configuration register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30154_H$ | GTM0MCS02EIRQEN | MCS Channel x error interrupt enable register 2 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30180_H$ | GTM0MCS03R0 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R0. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30184_H$ | GTM0MCS03R1 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R1. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30188_H$ | GTM0MCS03R2 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R2. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3018C_H$ | GTM0MCS03R3 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R3. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30190_H$ | GTM0MCS03R4 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R4. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30194_H$ | GTM0MCS03R5 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R5. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30198_H$ | GTM0MCS03R6 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R6. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3019C_H$ | GTM0MCS03R7 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R7. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $301A0_H$ | GTM0MCS03CTRL | MCS Channel control register 3. Most bits mirror the internal task specific register STA. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $301A4_H$ | GTM0MCS03ACB | MCS Channel ACB register 3. The Register mirrors the internal task specific register ACB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (28/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 301BC$_H$ | GTM0MCS03MHB | Memory High Byte register 3. The Register mirrors the internal task specific register MHB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 301C0$_H$ | GTM0MCS03PC | MCS Channel Program counter register 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 301C4$_H$ | GTM0MCS03IRQNOTIFY | MCS Channel x interrupt notification register 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 301C8$_H$ | GTM0MCS03IRQEN | MCS Channel x interrupt enable register 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 301CC$_H$ | GTM0MCS03IRQFORCINT | MCS Channel x software interrupt generation register 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 301D0$_H$ | GTM0MCS03IRQMODE | IRQ mode configuration register3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 301D4$_H$ | GTM0MCS03EIRQEN | MCS Channel x error interrupt enable register 3 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30200$_H$ | GTM0MCS04R0 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R0. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30204$_H$ | GTM0MCS04R1 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R1. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30208$_H$ | GTM0MCS04R2 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R2. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3020C$_H$ | GTM0MCS04R3 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R3. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30210$_H$ | GTM0MCS04R4 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R4. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30214$_H$ | GTM0MCS04R5 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R5. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30218$_H$ | GTM0MCS04R6 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R6. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3021C$_H$ | GTM0MCS04R7 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R7. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30220$_H$ | GTM0MCS04CTRL | MCS Channel control register 4. Most bits mirror the internal task specific register STA. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30224$_H$ | GTM0MCS04ACB | MCS Channel ACB register 4. The Register mirrors the internal task specific register ACB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3023C$_H$ | GTM0MCS04MHB | Memory High Byte register 4. The Register mirrors the internal task specific register MHB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30240$_H$ | GTM0MCS04PC | MCS Channel Program counter register 4 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30244$_H$ | GTM0MCS04IRQNOTIFY | MCS Channel x interrupt notification register 4 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30248$_H$ | GTM0MCS04IRQEN | MCS Channel x interrupt enable register 4 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3024C$_H$ | GTM0MCS04IRQFORCINT | MCS Channel x software interrupt generation register 4 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30250$_H$ | GTM0MCS04IRQMODE | IRQ mode configuration register 4 | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (29/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + $30254_H$ | GTM0MCS04EIRQEN | MCS Channel x error interrupt enable register 4 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30280_H$ | GTM0MCS05R0 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R0. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30284_H$ | GTM0MCS05R1 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R1. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30288_H$ | GTM0MCS05R2 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R2. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3028C_H$ | GTM0MCS05R3 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R3. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30290_H$ | GTM0MCS05R4 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R4. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30294_H$ | GTM0MCS05R5 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R5. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30298_H$ | GTM0MCS05R6 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R6. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3029C_H$ | GTM0MCS05R7 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R7. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $302A0_H$ | GTM0MCS05CTRL | MCS Channel control register 5. Most bits mirror the internal task specific register STA. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $302A4_H$ | GTM0MCS05ACB | MCS Channel ACB register 5. The Register mirrors the internal task specific register ACB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $302BC_H$ | GTM0MCS05MHB | Memory High Byte register 5. The Register mirrors the internal task specific register MHB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $302C0_H$ | GTM0MCS05PC | MCS Channel Program counter register 5 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $302C4_H$ | GTM0MCS05IRQNOTIFY | MCS Channel x interrupt notification register 5 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $302C8_H$ | GTM0MCS05IRQEN | MCS Channel x interrupt enable register 5 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $302CC_H$ | GTM0MCS05IRQFORCINT | MCS Channel x software interrupt generation register 5 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $302D0_H$ | GTM0MCS05IRQMODE | IRQ mode configuration register 5 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $302D4_H$ | GTM0MCS05EIRQEN | MCS Channel x error interrupt enable register 5 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30300_H$ | GTM0MCS06R0 | MCS Channel GPR6 registers. These registers mirror the internal task specific registers R0. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30304_H$ | GTM0MCS06R1 | MCS Channel GPR6 registers. These registers mirror the internal task specific registers R1. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30308_H$ | GTM0MCS06R2 | MCS Channel GPR6 registers. These registers mirror the internal task specific registers R2. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3030C_H$ | GTM0MCS06R3 | MCS Channel GPR6 registers. These registers mirror the internal task specific registers R3. | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (30/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + $30310_H$ | GTM0MCS06R4 | MCS Channel GPR6 registers. These registers mirror the internal task specific registers R4. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30314_H$ | GTM0MCS06R5 | MCS Channel GPR6 registers. These registers mirror the internal task specific registers R5. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30318_H$ | GTM0MCS06R6 | MCS Channel GPR6 registers. These registers mirror the internal task specific registers R6. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3031C_H$ | GTM0MCS06R7 | MCS Channel GPR6 registers. These registers mirror the internal task specific registers R7. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30320_H$ | GTM0MCS06CTRL | MCS Channel control register 6. Most bits mirror the internal task specific register STA. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30324_H$ | GTM0MCS06ACB | MCS Channel ACB register 6. The Register mirrors the internal task specific register ACB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3033C_H$ | GTM0MCS06MHB | Memory High Byte register 6. The Register mirrors the internal task specific register MHB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30340_H$ | GTM0MCS06PC | MCS Channel Program counter register 6 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30344_H$ | GTM0MCS06IRQNOTIFY | MCS Channel x interrupt notification register 6 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30348_H$ | GTM0MCS06IRQEN | MCS Channel x interrupt enable register 6 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3034C_H$ | GTM0MCS06IRQFORCINT | MCS Channel x software interrupt generation register 6 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30350_H$ | GTM0MCS06IRQMODE | IRQ mode configuration register 6 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30354_H$ | GTM0MCS06EIRQEN | MCS Channel x error interrupt enable register 6 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30380_H$ | GTM0MCS07R0 | MCS Channel GPR7 registers. These registers mirror the internal task specific registers R0. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30384_H$ | GTM0MCS07R1 | MCS Channel GPR7 registers. These registers mirror the internal task specific registers R1. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30388_H$ | GTM0MCS07R2 | MCS Channel GPR7 registers. These registers mirror the internal task specific registers R2. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3038C_H$ | GTM0MCS07R3 | MCS Channel GPR7 registers. These registers mirror the internal task specific registers R3. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30390_H$ | GTM0MCS07R4 | MCS Channel GPR7 registers. These registers mirror the internal task specific registers R4. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30394_H$ | GTM0MCS07R5 | MCS Channel GPR7 registers. These registers mirror the internal task specific registers R5. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30398_H$ | GTM0MCS07R6 | MCS Channel GPR7 registers. These registers mirror the internal task specific registers R6. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3039C_H$ | GTM0MCS07R7 | MCS Channel GPR7 registers. These registers mirror the internal task specific registers R7. | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (31/37)

| Module name | Address | Symbol | Register Name | Access Width | Support 207 | Support 208 | Access Protection PBG | Access Protection Other |
|---|---|---|---|---|---|---|---|---|
| GTM0 | <GTM0_base> + 303A0$_H$ | GTM0MCS07CTRL | MCS Channel control register 7. Most bits mirror the internal task specific register STA. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 303A4$_H$ | GTM0MCS07ACB | MCS Channel ACB register 7. The Register mirrors the internal task specific register ACB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 303BC$_H$ | GTM0MCS07MHB | Memory High Byte register 7. The Register mirrors the internal task specific register MHB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 303C0$_H$ | GTM0MCS07PC | MCS Channel Program counter register 7 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 303C4$_H$ | GTM0MCS07IRQNOTIFY | MCS Channel x interrupt notification register 7 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 303C8$_H$ | GTM0MCS07IRQEN | MCS Channel x interrupt enable register 7 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 303CC$_H$ | GTM0MCS07IRQFORCINT | MCS Channel x software interrupt generation register 7 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 303D0$_H$ | GTM0MCS07IRQMODE | IRQ mode configuration register 7 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 303D4$_H$ | GTM0MCS07EIRQEN | MCS Channel x error interrupt enable register 7 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30400$_H$ | GTM0MCS08R0 | MCS Channel GPR8 registers. These registers mirror the internal task specific registers R0. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30404$_H$ | GTM0MCS08R1 | MCS Channel GPR8 registers. These registers mirror the internal task specific registers R1. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30408$_H$ | GTM0MCS08R2 | MCS Channel GPR8 registers. These registers mirror the internal task specific registers R2. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3040C$_H$ | GTM0MCS08R3 | MCS Channel GPR8 registers. These registers mirror the internal task specific registers R3. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30410$_H$ | GTM0MCS08R4 | MCS Channel GPR8 registers. These registers mirror the internal task specific registers R4. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30414$_H$ | GTM0MCS08R5 | MCS Channel GPR8 registers. These registers mirror the internal task specific registers R5. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30418$_H$ | GTM0MCS08R6 | MCS Channel GPR8 registers. These registers mirror the internal task specific registers R6. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3041C$_H$ | GTM0MCS08R7 | MCS Channel GPR8 registers. These registers mirror the internal task specific registers R7. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30420$_H$ | GTM0MCS08CTRL | MCS Channel control register 8. Most bits mirror the internal task specific register STA. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30424$_H$ | GTM0MCS08ACB | MCS Channel ACB register 8. The Register mirrors the internal task specific register ACB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3043C$_H$ | GTM0MCS08MHB | Memory High Byte register 8. The Register mirrors the internal task specific register MHB. | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30440$_H$ | GTM0MCS08PC | MCS Channel Program counter register 8 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 30444$_H$ | GTM0MCS08IRQNOTIFY | MCS Channel x interrupt notification register 8 | 32 | √ | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (32/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + $30448_H$ | GTM0MCS08IRQEN | MCS Channel x interrupt enable register 8 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3044C_H$ | GTM0MCS08IRQFORCINT | MCS Channel x software interrupt generation register 8 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30450_H$ | GTM0MCS08IRQMODE | IRQ mode configuration register 8 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $30454_H$ | GTM0MCS08EIRQEN | MCS Channel x error interrupt enable register 8 | 32 | √ | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31000_H$ | GTM0MCS10R0 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R0. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31004_H$ | GTM0MCS10R1 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R1. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31008_H$ | GTM0MCS10R2 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R2. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3100C_H$ | GTM0MCS10R3 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R3. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31010_H$ | GTM0MCS10R4 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R4. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31014_H$ | GTM0MCS10R5 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R5. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31018_H$ | GTM0MCS10R6 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R6. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3101C_H$ | GTM0MCS10R7 | MCS Channel GPR0 registers. These registers mirror the internal task specific registers R7. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31020_H$ | GTM0MCS10CTRL | MCS Channel control register 0. Most bits mirror the internal task specific register STA. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31024_H$ | GTM0MCS10ACB | MCS Channel ACB register 0. The Register mirrors the internal task specific register ACB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31028_H$ | GTM0MCS1CTRG | MCS Clear trigger control register. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3102C_H$ | GTM0MCS1STRG | MCS Set trigger control register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3103C_H$ | GTM0MCS10MHB | Memory High Byte register 0. The Register mirrors the internal task specific register MHB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31040_H$ | GTM0MCS10PC | MCS Channel Program counter register 0 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31044_H$ | GTM0MCS10IRQNOTIFY | MCS Channel x interrupt notification register 0 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31048_H$ | GTM0MCS10IRQEN | MCS Channel x interrupt enable register 0 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3104C_H$ | GTM0MCS10IRQFORCINT | MCS Channel x software interrupt generation register 0 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31050_H$ | GTM0MCS10IRQMODE | IRQ mode configuration register 0 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31054_H$ | GTM0MCS10EIRQEN | MCS Channel x error interrupt enable register 0 | 32 | — | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (33/37)

| Module name | Address | Symbol | Register Name | Access Width | Support 207 | Support 208 | Access Protection PBG | Access Protection Other |
|---|---|---|---|---|---|---|---|---|
| GTM0 | <GTM0_base> + $31064_H$ | GTM0MCS1CTRLSTAT | MCS Control and Status register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31068_H$ | GTM0MCS1RESET | MCS Channel reset register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3106C_H$ | GTM0MCS1CAT | Cancel ARU transfer register. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31070_H$ | GTM0MCS1CWT | Cancel WURM instruction. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3107C_H$ | GTM0MCS1ERR | MCS Error register | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31080_H$ | GTM0MCS11R0 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R0. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31084_H$ | GTM0MCS11R1 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R1. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31088_H$ | GTM0MCS11R2 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R2. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3108C_H$ | GTM0MCS11R3 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R3. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31090_H$ | GTM0MCS11R4 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R4. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31094_H$ | GTM0MCS11R5 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R5. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31098_H$ | GTM0MCS11R6 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R6. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3109C_H$ | GTM0MCS11R7 | MCS Channel GPR1 registers. These registers mirror the internal task specific registers R7. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $310A0_H$ | GTM0MCS11CTRL | MCS Channel control register 1. Most bits mirror the internal task specific register STA. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $310A4_H$ | GTM0MCS11ACB | MCS Channel ACB register 1. The Register mirrors the internal task specific register ACB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $310BC_H$ | GTM0MCS11MHB | Memory High Byte register 1. The Register mirrors the internal task specific register MHB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $310C0_H$ | GTM0MCS11PC | MCS Channel Program counter register 1 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $310C4_H$ | GTM0MCS11IRQNOTIFY | MCS Channel x interrupt notification register 1 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $310C8_H$ | GTM0MCS11IRQEN | MCS Channel x interrupt enable register 1 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $310CC_H$ | GTM0MCS11IRQFORCINT | MCS Channel x software interrupt generation register 1 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $310D0_H$ | GTM0MCS11IRQMODE | IRQ mode configuration register 1 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $310D4_H$ | GTM0MCS11EIRQEN | MCS Channel x error interrupt enable register 1 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31100_H$ | GTM0MCS12R0 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R0. | 32 | — | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (34/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 31104$_H$ | GTM0MCS12R1 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R1. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31108$_H$ | GTM0MCS12R2 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R2. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3110C$_H$ | GTM0MCS12R3 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R3. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31110$_H$ | GTM0MCS12R4 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R4. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31114$_H$ | GTM0MCS12R5 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R5. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31118$_H$ | GTM0MCS12R6 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R6. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3111C$_H$ | GTM0MCS12R7 | MCS Channel GPR2 registers. These registers mirror the internal task specific registers R7. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31120$_H$ | GTM0MCS12CTRL | MCS Channel control register 2. Most bits mirror the internal task specific register STA. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31124$_H$ | GTM0MCS12ACB | MCS Channel ACB register 2. The Register mirrors the internal task specific register ACB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3113C$_H$ | GTM0MCS12MHB | Memory High Byte register 2. The Register mirrors the internal task specific register MHB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31140$_H$ | GTM0MCS12PC | MCS Channel Program counter register 2 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31144$_H$ | GTM0MCS12IRQNOTIFY | MCS Channel x interrupt notification register 2 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31148$_H$ | GTM0MCS12IRQEN | MCS Channel x interrupt enable register 2 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3114C$_H$ | GTM0MCS12IRQFORCINT | MCS Channel x software interrupt generation register 2 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31150$_H$ | GTM0MCS12IRQMODE | IRQ mode configuration register 2 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31154$_H$ | GTM0MCS12EIRQEN | MCS Channel x error interrupt enable register 2 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31180$_H$ | GTM0MCS13R0 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R0. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31184$_H$ | GTM0MCS13R1 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R1. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31188$_H$ | GTM0MCS13R2 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R2. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3118C$_H$ | GTM0MCS13R3 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R3. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31190$_H$ | GTM0MCS13R4 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R4. | 32 | — | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (35/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + 31194$_H$ | GTM0MCS13R5 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R5. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31198$_H$ | GTM0MCS13R6 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R6. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3119C$_H$ | GTM0MCS13R7 | MCS Channel GPR3 registers. These registers mirror the internal task specific registers R7. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 311A0$_H$ | GTM0MCS13CTRL | MCS Channel control register 3. Most bits mirror the internal task specific register STA. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 311A4$_H$ | GTM0MCS13ACB | MCS Channel ACB register 3. The Register mirrors the internal task specific register ACB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 311BC$_H$ | GTM0MCS13MHB | Memory High Byte register 3. The Register mirrors the internal task specific register MHB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 311C0$_H$ | GTM0MCS13PC | MCS Channel Program counter register 3 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 311C4$_H$ | GTM0MCS13IRQNOTIFY | MCS Channel x interrupt notification register 3 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 311C8$_H$ | GTM0MCS13IRQEN | MCS Channel x interrupt enable register 3 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 311CC$_H$ | GTM0MCS13IRQFORCINT | MCS Channel x software interrupt generation register 3 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 311D0$_H$ | GTM0MCS13IRQMODE | IRQ mode configuration register 3 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 311D4$_H$ | GTM0MCS13EIRQEN | MCS Channel x error interrupt enable register 3 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31200$_H$ | GTM0MCS14R0 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R0. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31204$_H$ | GTM0MCS14R1 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R1. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31208$_H$ | GTM0MCS14R2 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R2. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3120C$_H$ | GTM0MCS14R3 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R3. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31210$_H$ | GTM0MCS14R4 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R4. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31214$_H$ | GTM0MCS14R5 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R5. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31218$_H$ | GTM0MCS14R6 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R6. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 3121C$_H$ | GTM0MCS14R7 | MCS Channel GPR4 registers. These registers mirror the internal task specific registers R7. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + 31220$_H$ | GTM0MCS14CTRL | MCS Channel control register 4. Most bits mirror the internal task specific register STA. | 32 | — | √ | PBG2. PG2-GTM0 | — |

Table 25.7 List of Registers (36/37)

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + $31224_H$ | GTM0MCS14ACB | MCS Channel ACB register 4. The Register mirrors the internal task specific register ACB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3123C_H$ | GTM0MCS14MHB | Memory High Byte register 4. The Register mirrors the internal task specific register MHB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31240_H$ | GTM0MCS14PC | MCS Channel Program counter register 4 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31244_H$ | GTM0MCS14IRQNOTIFY | MCS Channel x interrupt notification register 4 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31248_H$ | GTM0MCS14IRQEN | MCS Channel x interrupt enable register 4 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3124C_H$ | GTM0MCS14IRQFORCINT | MCS Channel x software interrupt generation register 4 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31250_H$ | GTM0MCS14IRQMODE | IRQ mode configuration register 4 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31254_H$ | GTM0MCS14EIRQEN | MCS Channel x error interrupt enable register 4 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31280_H$ | GTM0MCS15R0 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R0. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31284_H$ | GTM0MCS15R1 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R1. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31288_H$ | GTM0MCS15R2 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R2. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3128C_H$ | GTM0MCS15R3 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R3. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31290_H$ | GTM0MCS15R4 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R4. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31294_H$ | GTM0MCS15R5 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R5. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $31298_H$ | GTM0MCS15R6 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R6. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $3129C_H$ | GTM0MCS15R7 | MCS Channel GPR5 registers. These registers mirror the internal task specific registers R7. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $312A0_H$ | GTM0MCS15CTRL | MCS Channel control register 5. Most bits mirror the internal task specific register STA. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $312A4_H$ | GTM0MCS15ACB | MCS Channel ACB register 5. The Register mirrors the internal task specific register ACB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $312BC_H$ | GTM0MCS15MHB | Memory High Byte register 5. The Register mirrors the internal task specific register MHB. | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $312C0_H$ | GTM0MCS15PC | MCS Channel Program counter register 5 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $312C4_H$ | GTM0MCS15IRQNOTIFY | MCS Channel x interrupt notification register 5 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $312C8_H$ | GTM0MCS15IRQEN | MCS Channel x interrupt enable register 5 | 32 | — | √ | PBG2. PG2-GTM0 | — |

**Table 25.7 List of Registers (37/37)**

| Module name | Address | Symbol | Register Name | Access Width | Support | | Access Protection | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 207 | 208 | PBG | Other |
| GTM0 | <GTM0_base> + $312CC_H$ | GTM0MCS15IRQFORCINT | MCS Channel x software interrupt generation register 5 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $312D0_H$ | GTM0MCS15IRQMODE | IRQ mode configuration register 5 | 32 | — | √ | PBG2. PG2-GTM0 | — |
| GTM0 | <GTM0_base> + $312D4_H$ | GTM0MCS15EIRQEN | MCS Channel x error interrupt enable register 5 | 32 | — | √ | PBG2. PG2-GTM0 | — |

the access size of RAM has to be 32-bit wide.

**Table 25.8 Register Reset Condition**

| Register Name | Reset Condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| All registers | √ | √ | √ | √ | √ |

## 25.4 Operation

### 25.4.1 Peripheral Bus Access

Due to its complexity, the GTM is attached to a bus system as a slave unit so as to provide register and MCS RAM access through the AEI interface to the CPU software and DMA functions. Please note that an access to the GTM registers and RAM modules has to be always 32-bit wide.

GTM contains MCS RAM structures. These RAM structures are not shared with other functionality and can be accessed via the bus system in a certain address range. For details of the memory map, see **Section 25.18.5, Memory Address Ranges** or **Section 25.19.5, Memory Address Ranges**.

GTM is accessible by all cores with the same access speed except the ICUMC core. See "**Section 3, CPU System**" for more details on CPU to peripheral access speed.

### 25.4.2 Reset

GTM is resettable by a dedicated bit, which allows the software to set GTM in reset state without using external reset. See "**Section 8, Reset Controller**".

### 25.4.3 Interrupt Factors

GTM generates up to 180 interrupt factors, which are visible outside of the GTM and are connected to the Interrupt Controller. These are ICM output interrupts as described in **Section 25.15.3, ICM Interrupt Signals**. The exact number of interrupts and their source depends on the GTM configuration. For details of each device, see **Section 25.1.5**.

### 25.4.4 Connection to ADCF

GTM can trigger the AD conversion start. The signals, which should trigger the conversion, are connected to the ADCF module inside this device. As trigger sources following signals are used:

- TIM_IRQ, after ICM
- ATOM_OUT
- MCS_IRQ, after ICM

Signals specified above are multiplexed together and routed to the ADCF. Multiplication scheme is described in "Section 26, Peripheral Interconnect (PIC)". The GTM can trigger the AD groups 0 to 4. In "Section 26, Peripheral Interconnect (PIC)" more details regarding the ADCF trigger selection can also be found.

### 25.4.5 Connection from ADCF

Every group of the ADCF generates one conversion end interrupt. This signal can be used to trigger the TIM channel of the GTM to save the timestamp information. After that, the TIM can start the DMA to transfer ADCF group's channels conversion results and the timestamp information to the memory. The ADCF group conversion end signals are connected in one-to-one manner to the TIM. The input signal to the TIM has to be defined during configuration of the TIM, for example: TIM0 channel 0 - external input signal or ADCF converter group 0 signal. The scheme of the ADCF to TIM connections is shown below:

- ADCF0 group 0 end of conversion interrupt -> TIM0 input 0 or TIM1 input 0 exclusively
- ADCF0 group 1 end of conversion interrupt -> TIM0 input 1 or TIM1 input 1 exclusively
- ADCF0 group 2 end of conversion interrupt -> TIM0 input 2 or TIM1 input 2 exclusively
- ADCF0 group 3 end of conversion interrupt -> TIM0 input 3 or TIM1 input 3 exclusively
- ADCF0 group 4 end of conversion interrupt -> TIM0 input 4 or TIM1 input 4 exclusively
- ADCF1 group 0 end of conversion interrupt -> TIM1 input 0
- ADCF1 group 1 end of conversion interrupt -> TIM1 input 1
- ADCF1 group 2 end of conversion interrupt -> TIM1 input 2
- ADCF1 group 3 end of conversion interrupt -> TIM1 input 3
- ADCF1 group 4 end of conversion interrupt -> TIM1 input 4

Please refer also to the "Section 26, Peripheral Interconnect (PIC)" for more information.

### 25.4.6 Sub-Module

Every ATOM submodule is connected to a 4-channel DTM (Dead Time Module) inside of the GTM. For the first 4 channels of the ATOM submodule, the connection to the DTM is selectable by a runtime configuration of the GTM registers. See **Section 25.12, Dead Time Module (DTM)** for DTM submodule description.

### 25.4.7 Safety Mechanism

The RAMs are protected by an ECC function. The ECC function is accessible and testable. For more details regarding ECC function, see "**Section 33, RAM Modules**". An ECC error will also be signaled to the respective MCS submodule.

An output signal from ATOM submodule can be read back using TIM submodule using the same pin. This function is a part of the safety mechanism. Please therefore see "**Section 28, Functional Safety**" for more details. Refer also to "Section 26, Peripheral Interconnect (PIC)" for more details.

The DTM submodule output signals are set into Hi-Z state automatically in reaction to an external input signal on the ESO (Emergency Shut-off) pin or on the error signal from ECM. Please refer also to "Section 26, Peripheral Interconnect (PIC)" and "**Section 29, Error Control Module (ECM)**" for more details.

### 25.4.8 Procedure of Module Standby and Limited Reset

This module supports module standby and limited reset functions. Before these functions are enabled, all of the followings must be ensured:

- Disable output pins of ATOM
  - Write the following register
    - GTM0ATOMiAGCOUTENSTAT = $5555_H$
  - Read the following register and wait the value matching
    - GTM0ATOMiAGCOUTENSTAT == $0_H$
- Stop operation of TBU
  - Write the following register
    - GTM0TBUCHEN = $5_H$
  - Read the following register and wait the value matching
    - GTM0TBUCHEN == $0_H$
- Disable all interrupts of sub-modules
  - Write the following registers
    - GTM0GTMIRQEN = $0_H$
    - GTM0GTMEIRQEN = $0_H$
    - GTM0ARUIRQEN = $0_H$
    - GTM0TIMixIRQEN = $0_H$
    - GTM0TIMixEIRQEN = $0_H$
    - GTM0ATOMixIRQEN = $0_H$
    - GTM0MCSixIRQEN= $0_H$
    - GTM0MCSixEIRQEN = $0_H$
    - GTM0CMPIRQEN = $0_H$
    - GTM0CMPEIRQEN = $0_H$
- Stop operation of MCS
  - Write the following registers
    - GTM0MCSixCTRL = $0_H$
    - GTM0MCSiCAT = $1FF_H$
    - GTM0MCS0CWT = $1FF_H$
  - Read the following register and wait the value matching
    - GTM0MCSixCTRL[0] == $0_H$
- No peripheral access to GTM

# 25.5 Introduction

## 25.5.1 Overview

This document is the specification for the Generic Timer Module (GTM). It contains a module framework with submodules of different functionality. These submodules can be combined in a configurable manner to form a complex timer module that serves different application domains and different classes within one application domain. Because of this scalability and configurability the timer is called generic.

The scalability and configurability is reached with an architecture philosophy where dedicated hardware submodules are located around a central routing unit (called Advanced Routing Unit (ARU)). The ARU can connect the submodules in a flexible manner. The connectivity is software programmable and can be configured during runtime.

Nevertheless, the GTM-IP is designed to unload the CPU or a peripheral core from a high interrupt load. Most of the tasks inside the GTM-IP can run -once setup by an external CPU- independent and in parallel to the software. There may be special situations, where the CPU has to take action but the goal of the GTM design was to reduce these situations to a minimum.

The hardware submodules have dedicated functionalities, e.g. there are timer input modules where incoming signals can be captured and characterized together with a notion of time. By combination of several submodules through the ARU complex functions can be established. E.g. the signals characterized at an input module can be routed to a signal processing unit where an intermediate value about the incoming signal frequency can be calculated.

The modules that help to implement such complex functions are called *infrastructural components* further on. These components are present in all GTM variants. However, the number of these components may vary from device to device.

Other submodules have a more general architecture and can fulfil typical timer functions, e.g. there are MCS and IO modules. A third group of submodules is responsible for supporting the implementation of safety functions to fulfil a defined safety level. The module ICM is responsible for interrupt services and defines the fourth group.

Each GTM-IP is build up therefore with submodules coming from those four groups. The application class is defined by the amount of components of those submodules integrated into the implemented GTM-IP.

### 25.5.2 Document Structure

The structure of this document is motivated out of the aforementioned submodule classes. **Section 25.6, GTM Architecture** describes the dedicated GTM-IP implementation this specification is written for. It gives an overview about the implemented submodules.

The following **Section 25.6** up to **Section 25.9** deals with the so called infrastructural components for routing, clock management and common time base functions. **Section 25.10**, **Section 25.11** and **Section 25.12** describe the signal input and output modules while the following **Section 25.13** explains the signal processing and generation submodule. **Section 25.14** describes the memory configuration submodule. The next sections provide a detailed description of application specific and safety related modules like the CMP and MON submodules. **Section 25.16** describes a module that bundles several interrupts coming from the other submodules and connect them to the outside world.

These submodule groups are shown in the following table:

| Chapter | Submodule | Group |
|---|---|---|
| **25.7** | Advanced Routing Unit (ARU) | Infrastructural components |
| **25.8** | Clock Management Unit (CMU) | Infrastructural components |
| **25.9** | Time Base Unit (TBU) | Infrastructural components |
| **25.10** | Timer Input Module (TIM) | IO Modules |
| **25.11** | ARU-connected Timer Output Module (ATOM) | IO Modules |
| **25.12** | Dead Time Module (DTM) | IO Modules |
| **25.13** | Multi Channel Sequencer (MCS) | Signal generation and processing |
| **25.14** | Memory Configuration (MCFG) | Memory Configuration |
| **25.15** | Interrupt Concentrator Module (ICM) | Interrupt services |
| **25.16** | Output Compare Unit (CMP) | Safety features |
| **25.17** | Monitoring Unit (MON) | Safety features |

## 25.6 GTM Architecture

### 25.6.1 Overview

As already mentioned in **Section 25.5, Introduction** the GTM-IP forms a generic timer platform that serves different application domains and different classes within these application domains. Depending on these multiple requirements of application domains multiple device configurations with different count of submodules (i.e. ATOM, MCS, TIM, TOM) and different count of channel per submodule (if applicable) are possible. In this section as an example of possible device configurations the GTM-IP_208 realization is outlined. The architecture of the GTM-IP_208 is depicted in **Figure 25.2**. The device dependent configuration (i.e. the count of submodules) is listed in **Section 25.19, GTM Device 208**.

### 25.6.1.1 GTM Architecture Block Diagram

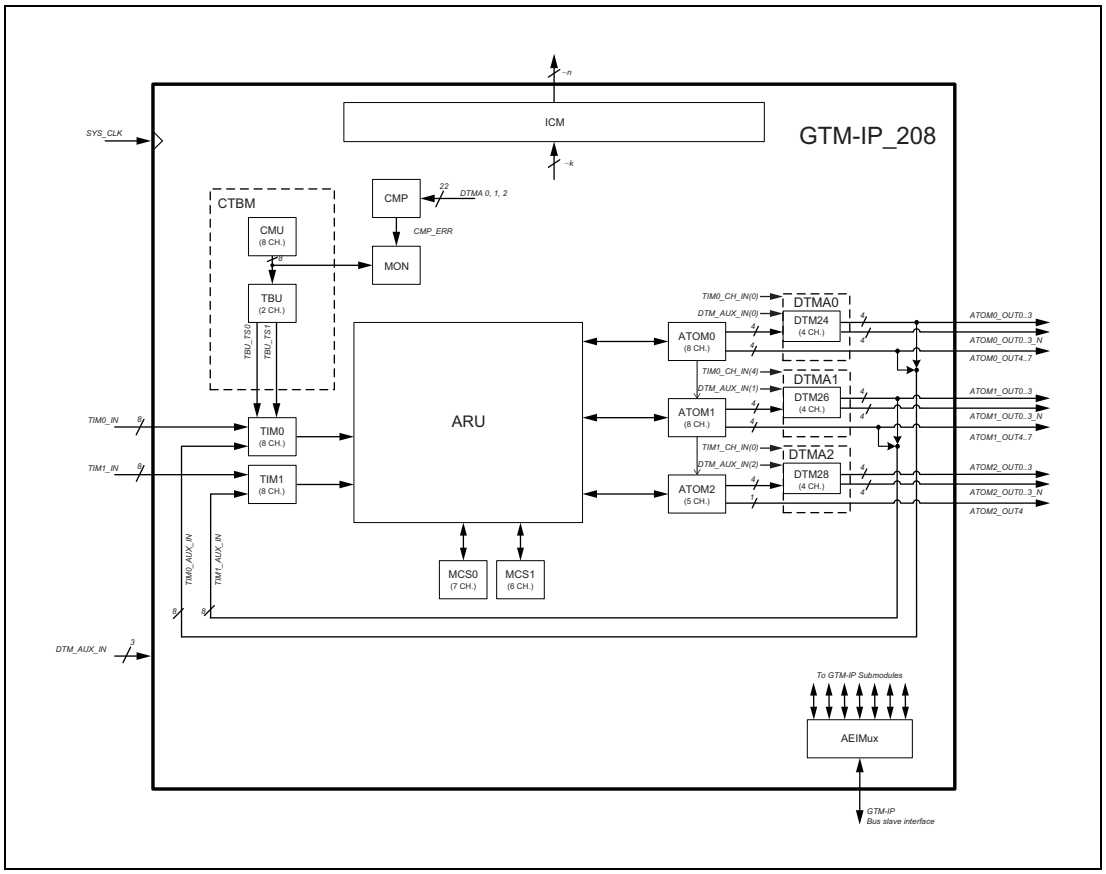



Figure 25.2 GTM Architecture Block Diagram

The central component of the GTM-IP is the Advanced Routing Unit (ARU) where most of the submodules are located around and connected to. This ARU forms together with the Clock Management Unit (CMU) and the Time Base Unit (TBU) the infrastructural part of the GTM. The ARU is able to route data from a connected source submodule to a connected destination submodule. The routing is done in a deterministic manner with a round-robin scheduling scheme of connected channels which receive data from ARU and with a worst case round-trip time.

The routed data word size of the ARU is 53 bit. The data word can logically be split into three parts. These parts are shown in **Figure 25.3, ARU Data Word** . Bits 0 to 23 and bits 24 to 47 typically hold data for the operation registers of the GTM-IP. This can be for example the duty cycle and period duration of a measured PWM input signal or the output characteristic of an output PWM to be generated. Another possible content of Data0 and Data1 can be two 24 bit values of the GTM-IP time bases TBU_TS0 and TBU_TS1. Bits 48 to 52 can contain control bits to send control information from one submodule to another. These ARU Control Bits (ACB) can have a different meaning for different submodules.

It is also possible to route data from a source to a destination and the destination can act later on as source for another destination. These routes through the GTM-IP are further on called data streams. For a detailed description of the ARU submodule see **Section 25.7, Advanced Routing Unit (ARU)** .

### 25.6.1.2 ARU Data Word Description

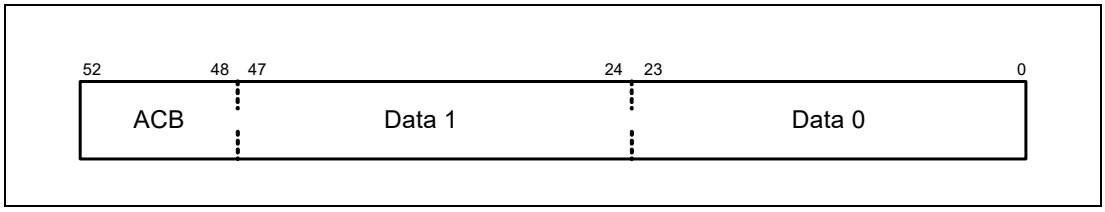



**Figure 25.3 ARU Data Word**

Signals are transferred into the GTM-IP at the Timer Input Modules (TIM). These modules are able to filter the input signals and annotate additional information. Each channel is for example able to measure pulse high or low times and the period of a PWM signal in parallel and route the values to ARU for further processing. The internal operation registers of the TIM submodule are 24 bits wide.

The Clock Management Unit (CMU) serves up to 8 different clocks for the GTM. It acts as a clock divider for the system clock. The counters implemented inside other submodules are typically driven from this submodule. Please note, that the CMU clocks are implemented as enable signals for the counters while the whole system runs with the GTM global clock SYS_CLK. This global clock typically corresponds to the microcontroller bus clock the GTM-IP is connected to and should not exceed 100MHz because of the power dissipation of the used transistors where the GTM is implemented with.

The TBU provides up to twice independent common time bases for the GTM-IP. In general, the number of time bases depends on the implemented device.

Signal outputs are generated with the ARU-connected TOMs (ATOM) and the corresponding Dead Time Modules (DTM).

The ATOMs offer the additional functionality to generate complex output signals without CPU interaction by serving these complex waveform characteristics by other submodules that are connected to the ARU like the Multi Channel Sequencer (MCS). The operation and shadow registers of the ATOM channels are 24 bit wide to have a higher resolution and to have the opportunity to compare against time base values coming from the TBU.

It is possible to trigger ATOM channels for a successor ATOM submodule through a trigger line between ATOM(x)_CH(7) and ATOM(x+1)_CH(0). But to avoid long trigger paths the GTM-IP integrator can configure after which ATOM submodule instance a register is placed into the trigger signal chain. Each register results in one SYS_CLK cycle delay of the trigger signal. See device specification of silicon vendor for unregistered trigger chain length.

Together with the MCS the ATOM is able to generate an arbitrary predefined output sequence at the GTM-IP output pins. The output sequence is defined by instructions located in RAM connected to the MCS submodule. The instructions define the points were an output signal should change or to react on other signal inputs. The output points can be one or two time stamps (or even angle stamp in case of an engine management system) provided by the TBU. Since the MCS is able to read data from the ARU it is also able to operate on incoming data routed from the TIM. Additionally, the MCS can process data that is located in its connected RAMs. The MCS RAM is located logically inside the MCS while the silicon vendor has to implement its own RAM technology there.

The two modules Compare Module (CMP) and Monitor Module (MON) implement safety related features. The CMP compares two output channels of an ATOM and sends the result to the MON submodule were the error is signalled to the CPU. The MON module is also able to monitor the ARU and CMU activities.

In the described implementation the submodules of the GTM-IP have a huge amount of different interrupt sources. These interrupt sources are grouped and concentrated by the Interrupt Concentrator Module (ICM) to form a much easier manageable bunch of interrupts that are visible outside of the GTM-IP.

On the GTM-IP top level there are some configurable signal connections from the signal output of the DTM modules to the input signals of the TIM modules.

The trigger out of TIM (i.e. the signals TIM[i]_EXT_CAPTURE(7:0) of each TIM instance i) are routed to ATOM instance [i] and TOM instance [i] with i=0..NTIM-1 (NTIM defines the number of available TIM instances, see **Table 25.2, Sub-Units and Channels**).

This TIM trigger can be used to trigger inside the ATOM or TOM instance either a channel or the global control register of AGC or TGC0/TGC1 unit.

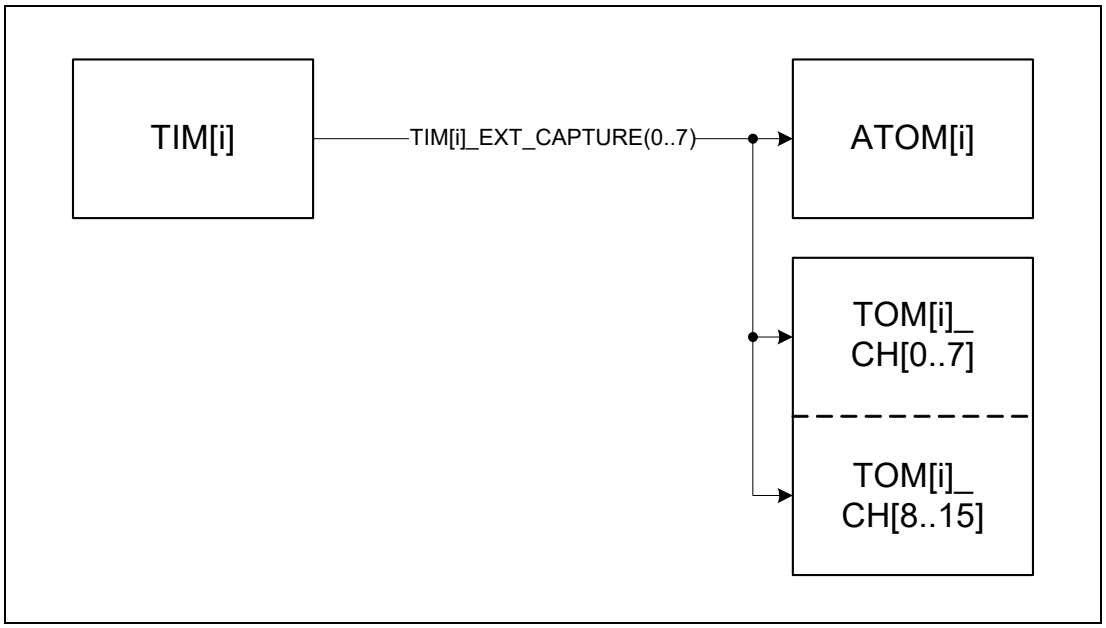



**Figure 25.4 The trigger out of TIM(1)**

The trigger out of TIM (i.e. the signals TIM[i]_EXT_CAPTURE(7:0) of each TIM instance i) are additionally routed to the MCS instance [i]. This trigger forwarding can be enabled by register GTM0GTMEXTCAPENi.

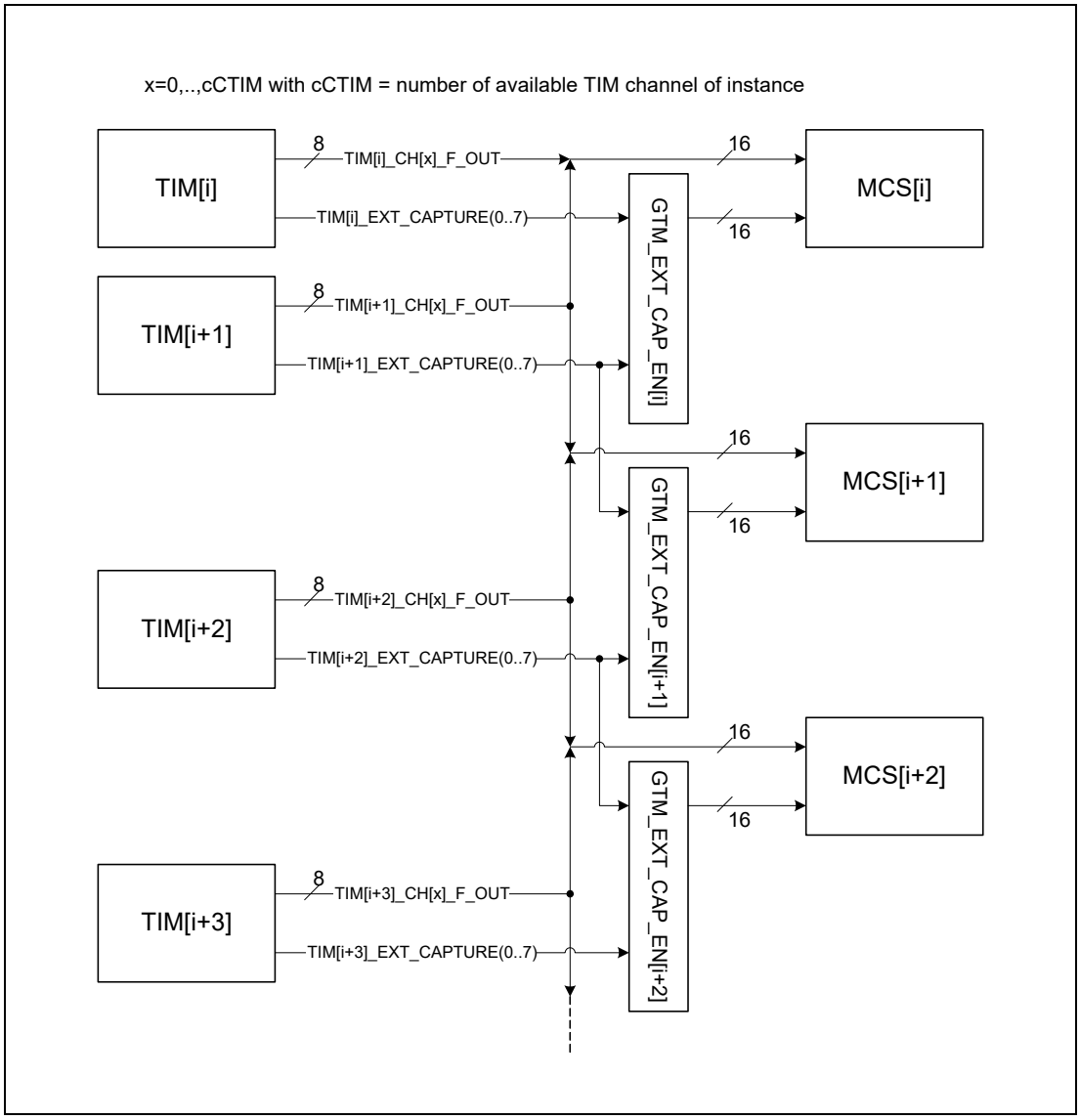



**Figure 25.5 The trigger out of TIM(2)**

## 25.6.2 GTM-IP Interfaces

In general the GTM-IP can be divided into four interface groups. Two interface groups represent the ports of the GTM-IP where incoming signals are assembled and outgoing signals are created. These interfaces are therefore connected to the GTM-IP input submodule TIM and to the GTM-IP output submodules DTM.

Another interface is the bus interface where the GTM-IP can be connected to the SoC system bus. This generic bus interface is described in more detail in **Section 25.6.2.1, GTM-IP Generic Bus Interface (AEI)** . The last interface is the interrupt controller interface. The GTM-IP provides several interrupt lines coming from the various submodules. These interrupt lines are concentrated inside the ICM and have to be adapted to the dedicated microcontroller environment where each interrupt handling can look different. The interrupt concept is described in more detail in **Section 25.6.5, GTM-IP Interrupt Concept** .

### 25.6.2.1 GTM-IP Generic Bus Interface (AEI)

The GTM-IP is equipped with a generic bus interface that can be widely adapted to different SoC bus systems. This generic bus interface is called AE-Interface (AEI). The adaptation of the AEI to SoC buses is typically done with a bridge module translating the AEI signals to the SoC bus signals of the silicon vendor. The AEI bus signals are depicted in the following table:

**Table 25.9 AEI bus signals**

| Signal name | I/O | Description | Bit width |
|---|---|---|---|
| AEI_SEL | I | GTM-IP select line | 1 |
| AEI_ADDR | I | GTM-IP address | 32 |
| AEI_PIPE | I | AEI Address phase signal | 1 |
| AEI_W1R0 | I | Read/Write access | 1 |
| AEI_WDATA | I | Write data bus | 32 |
| AEI_RDATA | O | Read data bus | 32 |
| AEI_READY | O | Data ready signal | 1 |
| AEI_STATUS | O | AEI Access status | 2 |

The AEI Status Signal may drive one of the following values:

**Table 25.10 AEI Status signal**

| AEI_STATUS | Description |
|---|---|
| 00 | No Error |
| 01 | Illegal Byte Addressing |
| 10 | Illegal Address Access |
| 11 | Unsupported Address |

The signal value “00” is driven if no error occurred during AEI access.

The signal value “01” is driven if the bus address is not an integer multiple of 4 (byte addressing).

The signal value “11” is driven if the address is not handled in the GTM.

The signal value “10” is driven if an illegal write access to one of the following register is performed:

(a) register is protected (e.g. protected by bit RF_PROT).

(b) writing a writable bit field of a register which is not implemented in the device with a value different from the reset value.

In case of an illegal write access signalled by status “10” the register will not be modified.

Reading registers will never return status “10”.

Write access to following addresses returns status "10" under special conditions:

GTM0ARUIRQFORCINT

GTM0ARUCADDREND

GTM0ATOMixCM0

GTM0ATOMixCM1

GTM0ATOMixSR0

GTM0ATOMixSR1

GTM0ATOMixRDADDR

GTM0ATOMixIRQFORCINT

GTM0CMPIRQFORCINT

GTM0CMUCLKEN

GTM0CMUGCLKNUM

GTM0CMUGCLKDEN

GTM0CMUCLKxCTRL

GTM0CMUGLBCTRL

GTM0GTMIRQFORCINT

GTM0GTMRST

GTM0TIMixCNTS

GTM0TIMixGPR1

GTM0TIMixIRQFORCINT

GTM0TIM00ECTRL

GTM0MCSixCTRL

GTM0MCSixPC

GTM0MCSixIRQFORCINT

GTM0MCSixCTRL

GTM0TBUCHEN

GTM0TBUxBASE

GTM0TBUxCTRL

MCS RAM during initialization

#### 25.6.2.2 GTM-IP Multi-master and multi-tasking support

To support multi-master and multi-task access to the registers of the GTM-IP a dedicated write-access scheme is used for critical control bits inside the IP that need such a mechanism. This can be for example a shared register where more than one channel can be controlled globally by one register write access. Such register bits are implemented inside the GTM-IP with a double bit mechanism, where the writing of ‘00’ and ‘11’ has no effect on the register bit and where ‘01’ sets the bit and ‘10’ resets the bit. If the CPU wants to read the status of the bit it always gets a ‘00’ if the bit is reset and it gets a ‘11’ if the bit is set.

### 25.6.3 ARU Routing Concept

One central concept of the GTM-IP is the routing mechanism of the ARU submodule for data streams. Each data word transferred between the ARU and its connected submodule is 53 bit wide. It is important to understand this concept in order to use the resources of the GTM-IP effectively. Each module that is connected to the ARU may provide an arbitrary number of ARU write channels and an arbitrary number of ARU read channels. In the following, the ARU write channels are named data sources and the ARU read channels are named data destinations.

The concept of the ARU intends to provide a flexible and resource efficient way for connecting any data source to an arbitrary data destination. In order to save resource costs, the ARU does not implement a switch matrix, but it implements a data router with serialized connectivity providing the same interconnection flexibility. **Figure 25.6** shows the ARU data routing principle. Data sources are marked with a green rectangle and the data destinations are marked with yellow rectangles. The dashed lines in the ARU depict the configurable connections between data sources and data destinations. A connection between a data source and a data destination is also called a data stream.

### 25.6.3.1 Principle of data routing using ARU

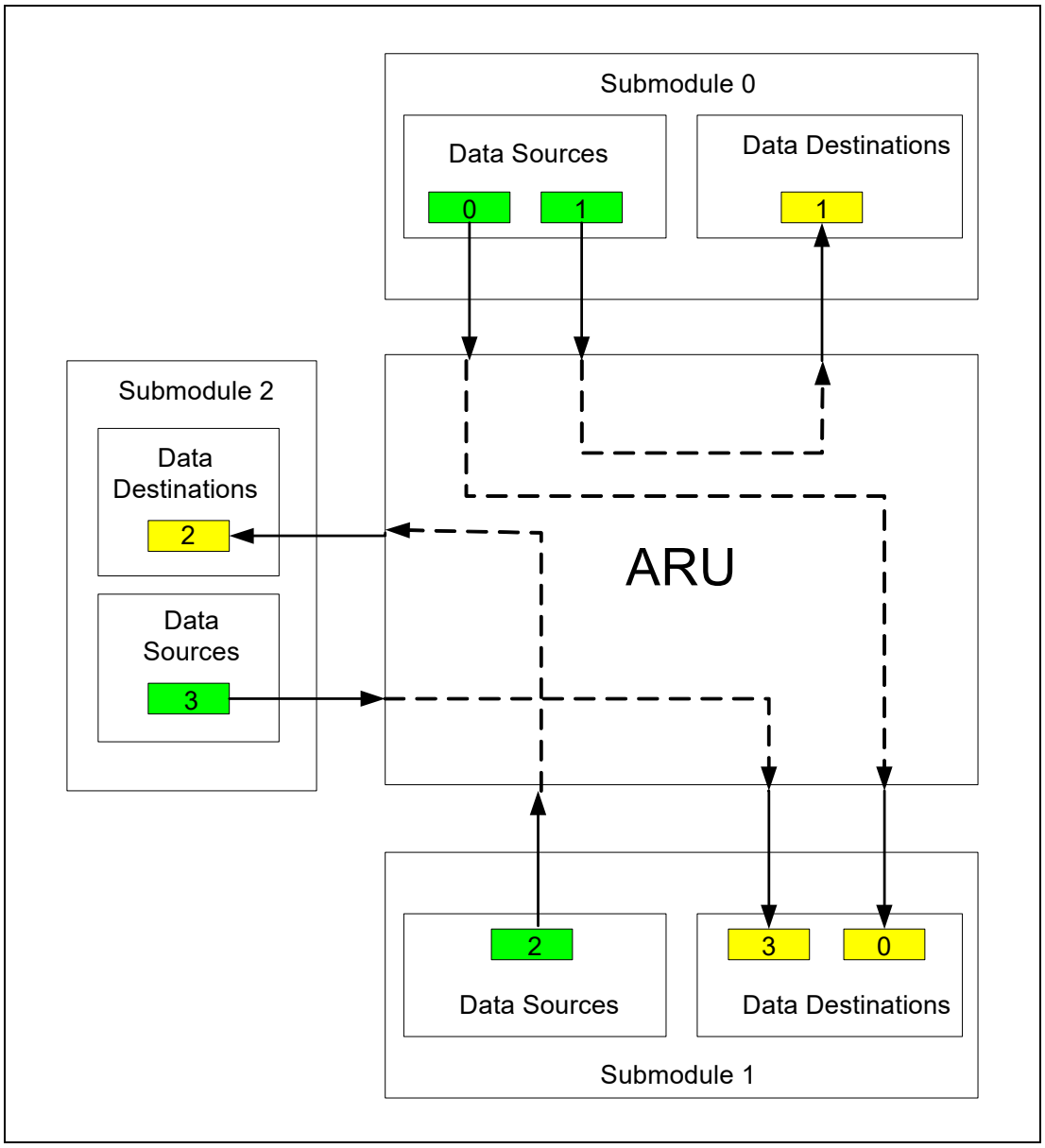



**Figure 25.6 Principle of data routing using ARU**

The configuration of the data streams is realized according to the following manner: Each data source has its fixed and unique source address: The fixed address of each data source is pointed out by the numbers in the green boxes of **Figure 25.6**. The address definitions of all available data sources in the GTM-IP can be obtained from **Table 25.204, ARU Write Address Overview**, **Table 25.211, ARU Write Address Overview**. The connection from a specific data source to a specific data destination is defined by configuring the corresponding address of a data source in the desired data destination. The configured address of each data destination is pointed out by the numbers in the yellow boxes of **Figure 25.6**.

Normally, the destination is idle and waits for data from the source. If the source offers new data, the destination does a destructive read, processes the data and goes idle again. The same data is never read twice.

The functionality of the ARU is as follows: The ARU sequentially polls the data destinations of the connected modules in a round-robin order. If a data destination requests new data from its configured data source and the data source has data available, the ARU delivers the data to the destination and it informs both, the data source and destination that the data is transferred. The data source marks the delivered ARU data as invalid which means that the destination consumed the data.

It should be noted that each data source should only be connected to a single data destination. This is because the destinations consume the data. If two destinations would reference the same source one destination would consume the data before the other destination could consume it. Since the data transfers are blocking, the second destination would block until it receives new data from the source. On the other hand, the transfer from a data source to the ARU is also blocking, which means that the source channel can only provide new data to the ARU when an old data word is consumed by a destination. In order to speed up the process of data transfers, the ARU handles two different data destinations in parallel.

Following table gives an overview about the number of channels for the GTM-IP_208 variant used as a reference within this chapter.

**Table 25.11 The number of channels for the GTM-IP_208 variant**

| Submodule | Number of data sources | Number of data destinations |
|---|---|---|
| ARU | 1 | 0 |
| CMU | 0 | 0 |
| TBU | 0 | 0 |
| TIM 0-1 | 16 | 0 |
| ATOM 0-2 | 21 | 21 |
| DTM | 0 | 0 |
| MCS 0-1 | 39 | 13 |
| ICM | 0 | 0 |
| CMP | 0 | 0 |
| MON | 0 | 0 |
| Total | 77 | 34 |

### 25.6.3.2 ARU Round Trip Time

The ARU uses a round-robin arbitration scheme with a fixed round trip time for all connected data destinations. This means that the time between two adjacent read requests resulting from a data destination channel always takes the round trip time, independently if the read request succeeds or fails.

#### 25.6.3.3 ARU Blocking Mechanism

Another important concept of the ARU is its blocking mechanism that is implemented for transferring data from a data source to a data destination. This mechanism is used by ARU connected submodules to synchronize the submodules to the routed data streams. **Figure 25.7** explains the blocking mechanism.

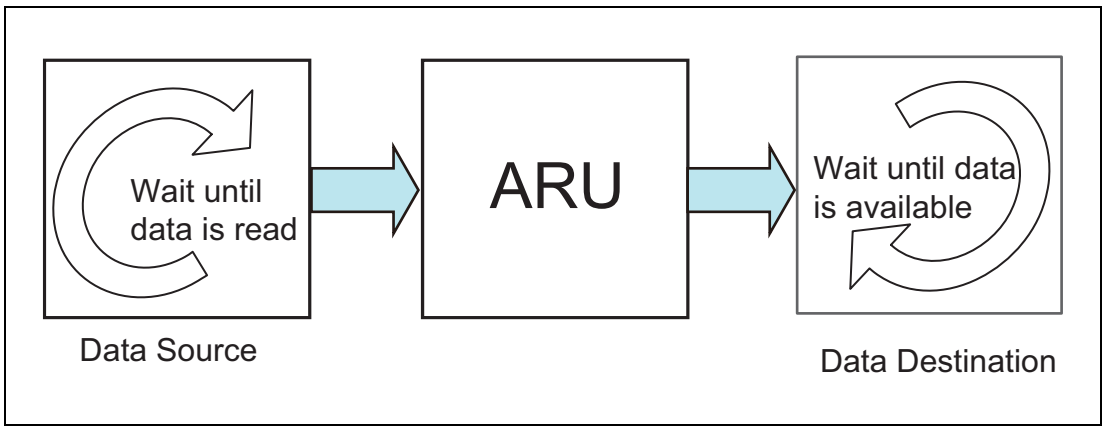



**Figure 25.7 Graphical representation of ARU blocking mechanism**

If a data destination requests data from a data source over the ARU but the data source does not have any data yet, it has to wait until the data source provides new data. In this case the submodule that owns the data destination may perform other tasks. When a data source produces new data faster than a data destination can consume the data the source raises an error interrupt and signals that the data could not be delivered in time. The new data is marked as valid for further transfers and the old data is overwritten.

In any case the round trip time for the ARU has a fixed reset value for a specific device configuration. The end value of the roundtrip counter can be changed with a configuration register GTM0ARUCADDREND inside the ARU. For more details see the ARU specific chapter.

Please refer also to **Section 25.18, GTM Device 207** and **Section 25.19, GTM Device 208**.

It is possible to reset the ARU roundtrip counter ARU_CADDR manually synchronous to CMU clock enable from configuration register inside CMU module. See CMU specific chapter for more details.

### 25.6.4 GTM-IP Clock and Time Base Management (CTBM)

Inside the GTM-IP several subunits are involved in the clock and time base management of the whole GTM. **Figure 25.8** shows the connections and sub blocks involved in these tasks. The sub blocks involved are called Clock and Time Base Management (CTBM) modules further on.

### 25.6.4.1 GTM-IP Clock and time base management architecture

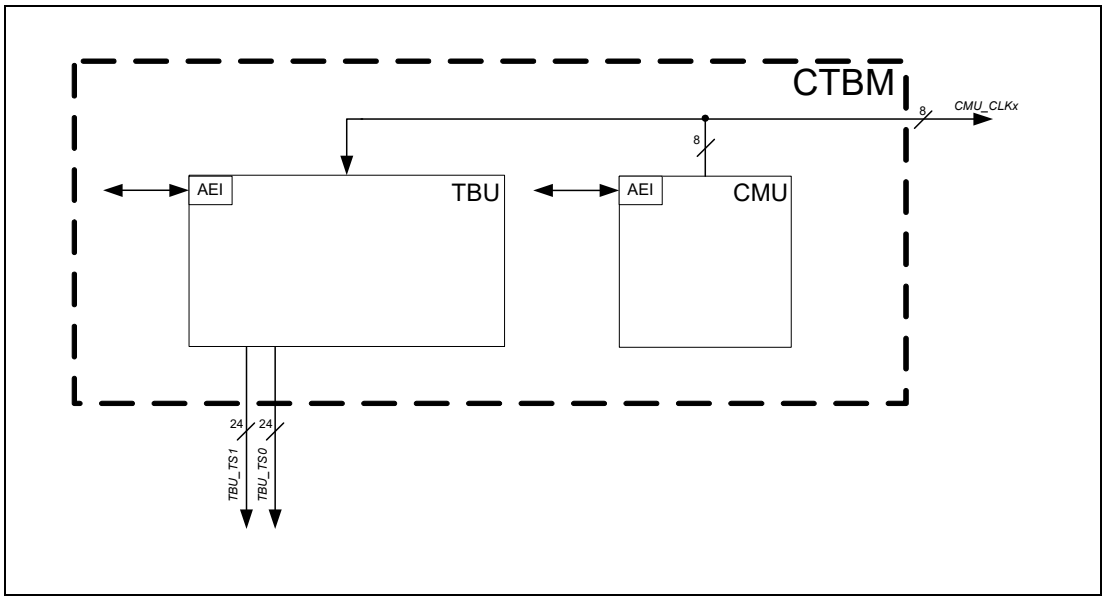



**Figure 25.8 GTM-IP Clock and time base management architecture**

One important module of the CTBM is the Clock Management Unit (CMU) which generates 8 clocks for the submodules of the GTM. For a detailed description of the CMU functionality and clocks see **Section 25.8, Clock Management Unit (CMU)** .

Inside the Time Base Unit (TBU) one of these eight clocks is used per channel to generate a common time base for the GTM. The TBU functionality is described in **Section 25.9, Time Base Unit (TBU)**.

In this device the TBU submodule generates the two time base signals TBU_TS0 and TBU_TS1 which are widely used inside the GTM as common time bases for signal characterization and generation.

### 25.6.5 GTM-IP Interrupt Concept

The submodules of the GTM-IP can generate thousands of interrupts on behalf of internal events. This high amount of interrupts is combined inside the Interrupt Concentrator Module (ICM) into interrupt groups. In this interrupt groups the GTM-IP submodule interrupt signals are bundled to a smaller set of interrupts. Out of these interrupt sets a smaller amount of interrupt signals is created and signalled outside of the GTM-IP as a signal GTM_<MOD>_IRQ, whereas <MOD> identifies the name of the corresponding GTM-IP submodule.

Moreover, each output signal GTM_<MOD>_IRQ has a corresponding input signal GTM_<MOD>_IRQ_CLR that can be used for clearing the interrupts. This input signals can be used by the surrounding microcontroller system as:

- acknowledge signal from a DMA controller
- validation signal from ADC
- clear signal from an GTM-external interrupt controller to do an atomic clear while entering an ISR routine

The controlling of the individual interrupts is done inside the submodules. If a submodule consists of several submodule channels that are most likely to work independent from each other (like TIM, MCS and ATOM), each submodule channel has its own interrupt control and status register set, named as interrupt set in the following. Other submodules (ARU, CMP and global GTM functionality) have a common interrupt set for the whole submodule.

The interrupt set consists of four registers: The IRQ_EN register, the IRQ_NOTIFY register, the IRQ_FORCINT register, and the IRQ_MODE register. While the registers IRQ_EN, IRQ_NOTIFY, and IRQ_FORCINT signalize the status and allow controlling of each individual interrupt source within an interrupt set, the register IRQ_MODE configures the interrupt mode that is applied to all interrupts that belong to the same interrupt set.

In order to support a wide variety of microcontroller architectures and interrupt systems with different interrupt signal output characteristics and internal interrupt handling the following four modes can be configured:

- Level mode
- Pulse mode
- Pulse-Notify mode
- Single-Pulse mode

These interrupt modes are described in more details in the following subsections.

The register IRQ_EN allows the enabling and disabling of an individual interrupt within an interrupt set. Independent of the configured mode, only enabled interrupts can signalize an interrupts on its signal GTM_<MOD>_IRQ.

The register IRQ_NOTIFY collects the occurrence of interrupt events. The behavior for setting a bit in this register depends on the configured mode and thus it is described later on in the mode descriptions.

Independent of the configured mode any write access with value ‘1’ to a bit in the register IRQ_NOTIFY always clears the corresponding IRQ_NOTIFY bit.

Moreover, the enabling of a disabled interrupt sources with a write access to the register IRQ_EN also clears the corresponding bit in the IRQ_NOTIFY register but only if the error interrupt source EIRQ_EN is disabled. However, if the enabling of a disabled interrupt is simultaneous to an incoming interrupt event, the interrupt event is dominant and the register IRQ_NOTIFY is not cleared.

Additionally, each write access to the register IRQ_MODE, clears all bits in the IRQ_NOTIFY register. It should be notified that the clearing of IRQ_NOTIFY is applied independently of the written data (e.g. no mode change).

Thus, a secure way for reconfiguring the interrupt mode of an interrupt set, is to disable all interrupts of the interrupt set with the register IRQ_EN, define the new interrupt mode by writing register IRQ_MODE, followed by enabling the desired interrupts with the register IRQ_EN.

Thus, a secure way for reconfiguring the interrupt mode of an error interrupt set, is to disable all error interrupts of the error interrupt set with the register EIRQ_EN, define the new interrupt mode by writing register IRQ_MODE, followed by enabling the desired error interrupts with the register EIRQ_EN.

The register IRQ_FORCINT is used by software for triggering individual interrupts with a write access with value ‘1’. Since a write access to IRQ_FORCINT only generates a single pulse, IRQ_FORCINT is not implemented as a true register and thus any read access to IRQ_FORCINT always results with a value of ‘0’.

It should be noted, that the mechanism for triggering interrupts with IRQ_FORCINT is globally disabled after reset. It has to be explicitly enabled by clearing the bit RF_PROT in the register GTM0GTMCTRL (see **Section 25.6.9.3, GTM0GTMCTRL** )

For the modules AEI-bridge, TIM, MCS and CMP each interrupt may configured to raise instead of the normal interrupt an error interrupt if enabled by the corresponding error interrupt enable bit in register EIRQ_EN.

Note, it is possible for one source to enable the normal interrupt and the error interrupt in parallel. Because of both interrupt clear signals could reset the notify bit this could cause problems in a system and therefore it is inadvisable.

Similar to enabling an interrupt, the enabling of a disabled error interrupt source with a write access to the register EIRQ_EN also clears the corresponding bit in the IRQ_NOTIFY register only if the interrupt source IRQ_EN is disabled. However, if the enabling of a disabled error interrupt is simultaneous to an incoming interrupt event, the interrupt event is dominant and the register IRQ_NOTIFY is not cleared.

All enabled error interrupts are or-combined inside the ICM and assigned to the dedicated GTM port gtm_err_irq. A corresponding input gtm_err_irq_clr allows the reset of this error interrupt from outside the GTM (hardware clear).

To be able to detect the module source of the error interrupt the ICM provides the register GTM0ICMIRQGMEI.

The error interrupt causing channel can be determined for the modules TIM by evaluating the ICM register GTM0ICMIRQGCEI1.

The error interrupt causing channel can be determined for MCS by evaluating the ICM register GTM0ICMIRQGCEI3.

### 25.6.5.1 Level interrupt mode

The default interrupt mode is the Level Interrupt Mode. In this mode each occurred interrupt event is collected in the register IRQ_NOTIFY, independent of the corresponding enable bit of register IRQ_EN and EIRQ_EN.

An interrupt event, which is defined as a pulse on the signal Int_out of **Figure 25.9** , may be triggered by the interrupt source of the submodule or by software performing a write access to the corresponding register IRQ_FORCINT, with a disabled bit RF_PROT in register GTM0GTMCTRL.

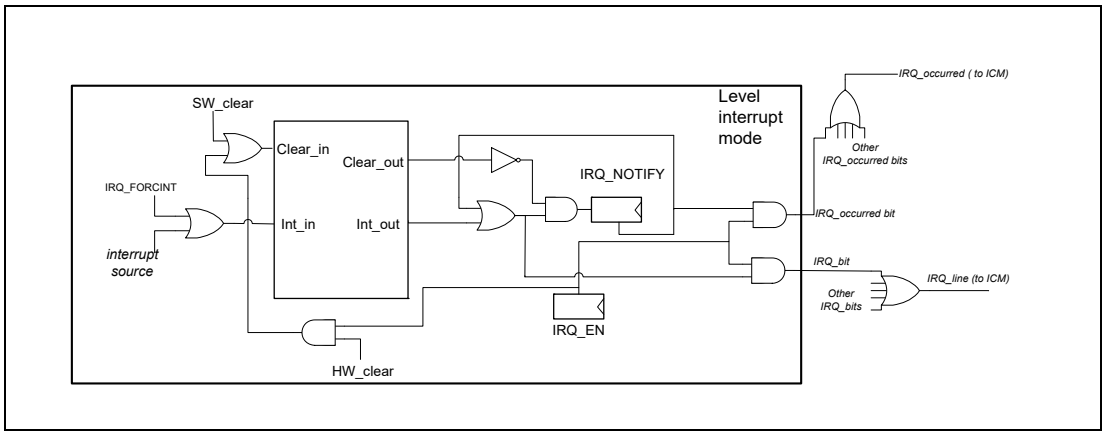



**Figure 25.9 Level interrupt mode scheme**

A collected interrupt bit in register IRQ_NOTIFY may be cleared by a clear event, which is defined as a pulse on signal Clear_out of **Figure 25.9**. A clear event can be performed with a write access with value '1' to the corresponding bit in the register IRQ_NOTIFY leading to a pulse on signals SW_clear. A clear event may also result from an externally connected signal GTM_<MOD>_IRQ_CLR, which is routed to the signal HW_clear of **Figure 25.9**. However, the hardware clear mechanism is only possible, if the corresponding interrupt is enabled by register IRQ_EN.

As the **Table 25.12** shows, interrupt events are dominant in the case of a simultaneous interrupt event and clear event.

**Table 25.12 Priority of Interrupt Events and Clear Events**

| Int_in | Clear_in | Int_out | Clear_out |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |

As it can be seen from the **Figure 25.9** an occurred interrupt event is signalled as a constant signal level with value 1 to the signal IRQ_bit, if the corresponding interrupt is enabled in register IRQ_EN.

With exception of the submodules ARU and DPLL, the signal IRQ_bit is OR-combined with the neighboring IRQ_bit signals of the same interrupt set and they are routed as a signal IRQ_line to the interrupt concentrator module (ICM). The interrupt signals IRQ_bit of the submodules DPLL and ARU are routed directly as a signal IRQ_line to the submodule ICM. In some cases (submodules TOM and ATOM) the ICM may further OR-combine several IRQ_line signals to an outgoing interrupt signal GTM_<MOD>_IRQ. In the other cases the IRQ_line signals are directly connected to the outgoing signals GTM_<MOD>_IRQ, within the submodule ICM.

The signal IRQ_occurred is connected in a similar way as the signal IRQ_line, however this signal is used for monitoring the interrupt state of the register IRQ_NOTIFY in the registers of the ICM.

The additional error interrupt enable mechanism for level interrupt is shown below.

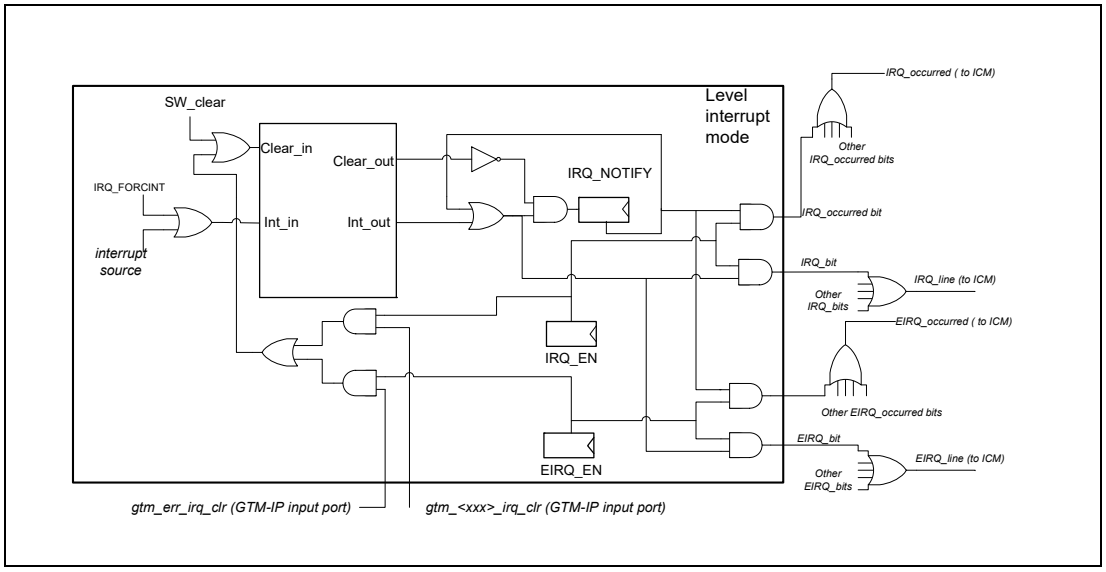



**Figure 25.10 Level interrupt scheme for modules AEI-bridge, TIM, MCS, CMP**

A collected interrupt bit in register IRQ_NOTIFY may be cleared by a clear event, which is defined as a pulse on signal Clear_out of **Figure 25.10**. A clear event can be performed with a write access with value '1' to the corresponding bit in the register IRQ_NOTIFY leading to a pulse on signals SW_clear. A clear event may also result from externally connected signal gtm_<MOD>_irq_clr or gtm_err_irq_clr, which is routed as a HW_clear to Clear_in of **Figure 25.10**. However, the hardware clear mechanism is only possible, if the corresponding interrupt or error interrupt is enabled by register IRQ_EN or EIRQ_EN.

As it can be seen from the **Figure 25.10** an occurred interrupt event is signalled as a constant signal level with value 1 to the signal IRQ_bit, if the corresponding interrupt is enabled in register IRQ_EN.

### 25.6.5.2 Pulse interrupt mode

The Pulse interrupt mode behavior can be observed from **Figure 25.11**.

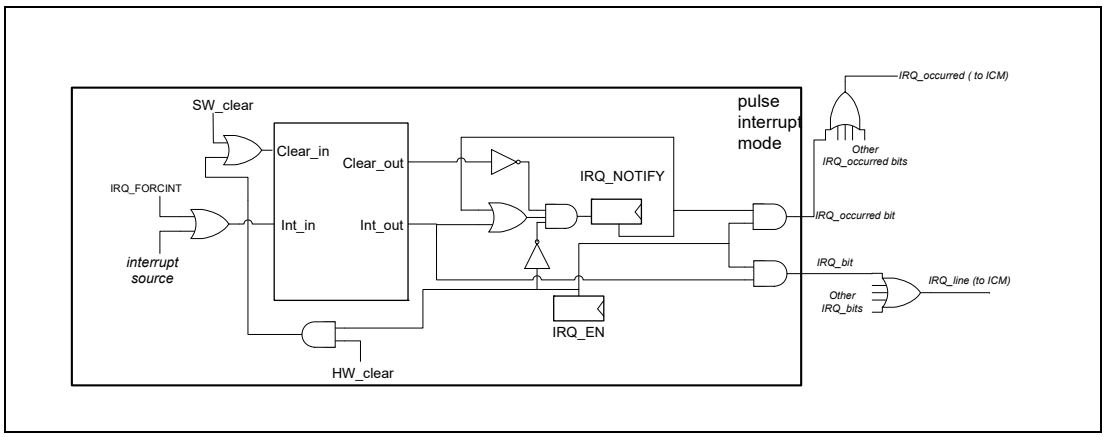



**Figure 25.11 Pulse interrupt mode scheme**

In Pulse Interrupt Mode each Interrupt Event will generate a pulse on the IRQ_bit signal if IRQ_EN is enabled.

As it can be seen from the figure, the interrupt bit in IRQ_NOTIFY register is always cleared if IRQ_EN is enabled.

However, if an interrupt is disabled in the register IRQ_EN, an occurred interrupt event is captured in the register IRQ_NOTIFY, in order to allow polling for disabled interrupts by software.

Disabled interrupts may be cleared by an interrupt clear event.

In Pulse interrupt mode, the signal IRQ_occurred is always 0.

The additional error interrupt enable mechanism for pulse interrupt is shown below.

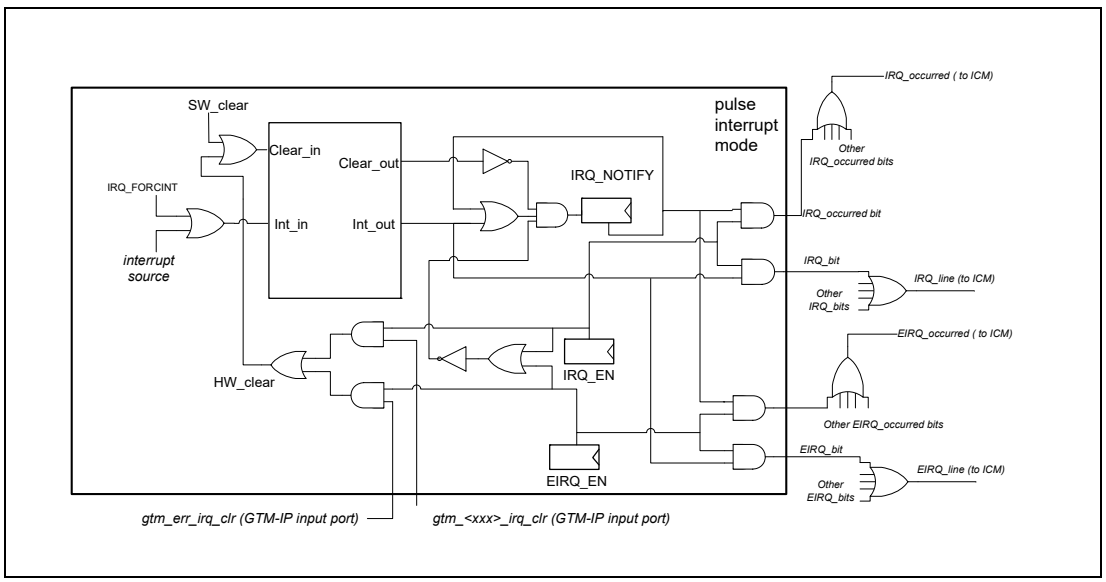



**Figure 25.12 Pulse interrupt scheme for modules AEI-bridge, TIM, MCS, CMP**

In Pulse Interrupt Mode each Interrupt Event will generate a pulse on the EIRQ_bit signal if EIRQ_EN is enabled.

As it can be seen from the figure, the interrupt bit in IRQ_NOTIFY register is always cleared if EIRQ_EN or IRQ_EN are enabled.

However, if an error interrupt is disabled in the register EIRQ_EN, an occurred error interrupt event is captured in the register IRQ_NOTIFY, in order to allow polling for disabled error interrupts by software.

Disabled error interrupts may be cleared by an error interrupt clear event.

In Pulse interrupt mode, the signal EIRQ_occurred is always 0.

### 25.6.5.3 Pulse-notify interrupt mode

In Pulse-notify Interrupt mode, all interrupt events are captured in the register IRQ_NOTIFY. If an interrupt is enabled by the register IRQ_EN, each interrupt event will also generate a pulse on the IRQ_bit signal. The signal IRQ_occurred will be high if interrupt is enabled in register IRQ_EN and the corresponding bit of register IRQ_NOTIFY is set. The Pulse-notify interrupt mode is shown in **Figure 25.13**.

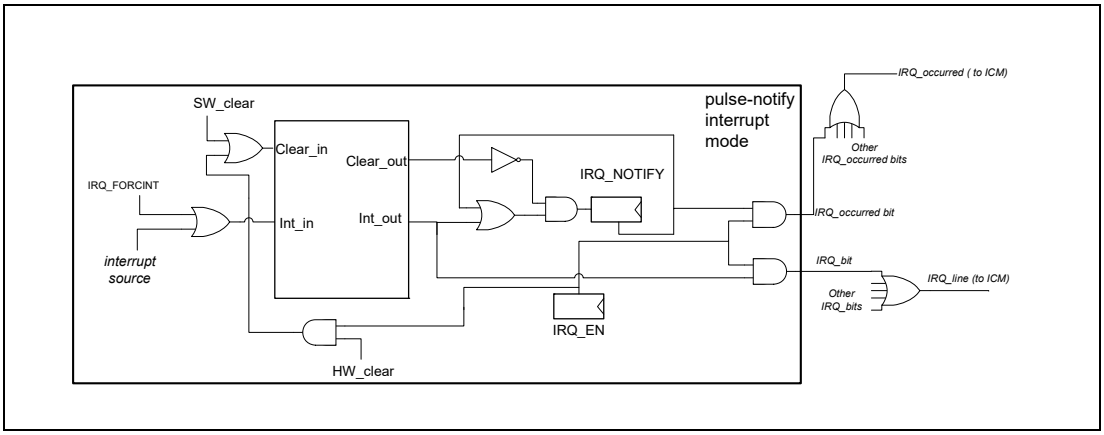



**Figure 25.13 Pulse-notify interrupt mode scheme**

The additional error interrupt enable mechanism for pulse-notify interrupt is shown below

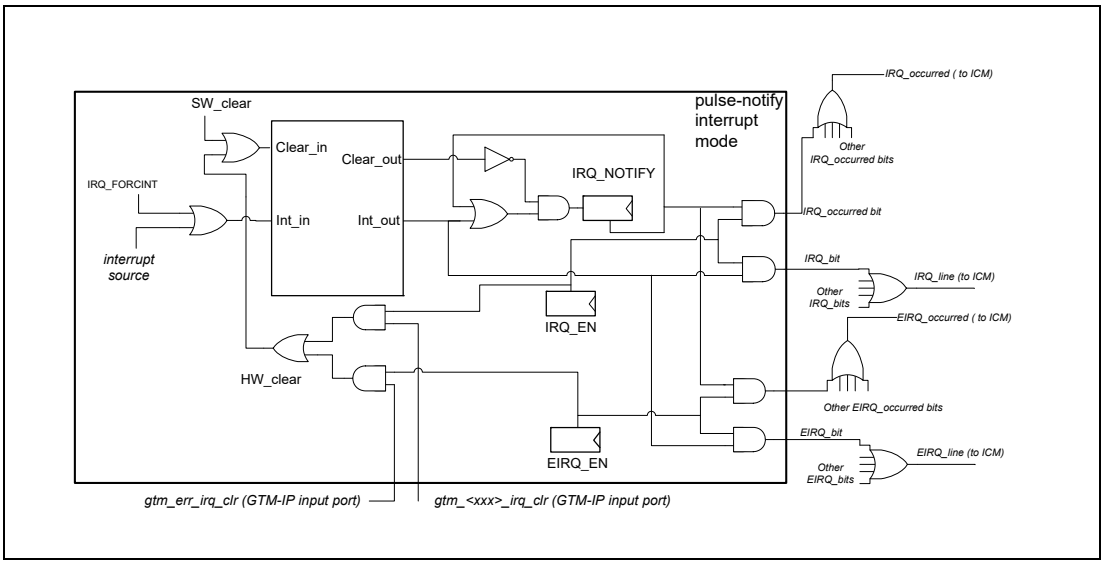



**Figure 25.14 Pulse-notify interrupt scheme for modules AEI-bridge, TIM, MCS, CMP**

In Pulse-notify Interrupt mode, all error interrupt events are captured in the register IRQ_NOTIFY. If an error interrupt is enabled by the register EIRQ_EN, each error interrupt event will also generate a pulse on the EIRQ_bit signal. The signal EIRQ_occurred will be high if error interrupt is enabled in register EIRQ_EN and the corresponding bit of register IRQ_NOTIFY is set. The Pulse-notify interrupt mode for error interrupts is shown in **Figure 25.14**.

### 25.6.5.4 Single-pulse interrupt mode

In Single-pulse Interrupt Mode, an interrupt event is always captured in the register IRQ_NOTIFY, independent of the state of IRQ_EN. However, only the first interrupt event of an enabled interrupt within a common interrupt set is forwarded to signal IRQ_line. Additional interrupt events of the same interrupt set cannot generate pulses on the signal IRQ_line, until the corresponding bits in register IRQ_NOTIFY of enabled interrupts are cleared by a clear event. The IRQ_occurred signal line will be high, if the IRQ_EN and the IRQ_NOTIFY register bits are set. The Single-pulse interrupt mode is shown in **Figure 25.15**.

The only exceptions are the modules ARU and DPLL. In these modules the IRQ_occurred bit of each interrupt is directly connected (without OR-conjunction of neighboring IRQ_occurred bits) to the inverter for suppressing further interrupt pulses.

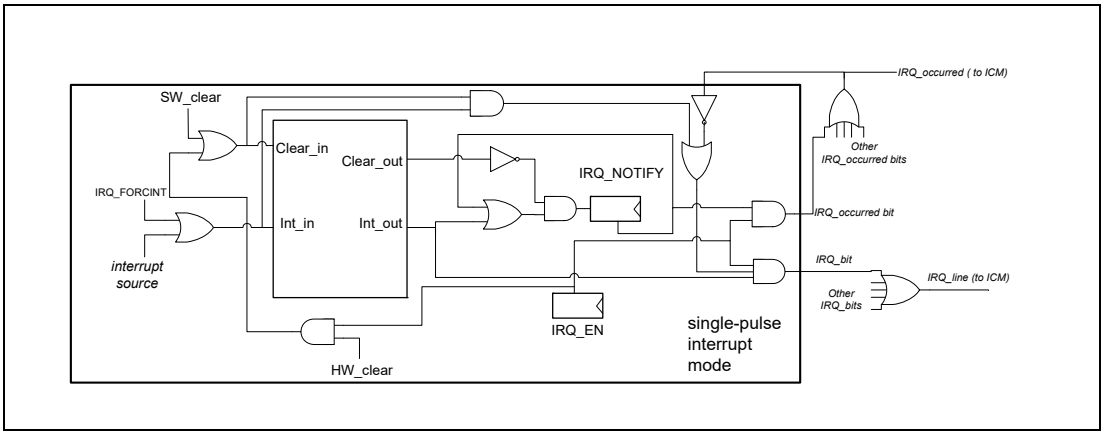



**Figure 25.15 Single-pulse interrupt mode scheme**

To avoid unexpected IRQ behavior in the single pulse mode, all desired interrupt sources should be enabled by a single write access to IRQ_EN and the notification bits should be cleared by a single write access to the register IRQ_NOTIFY.

The additional error interrupt enable mechanism for single-pulse interrupt is shown below

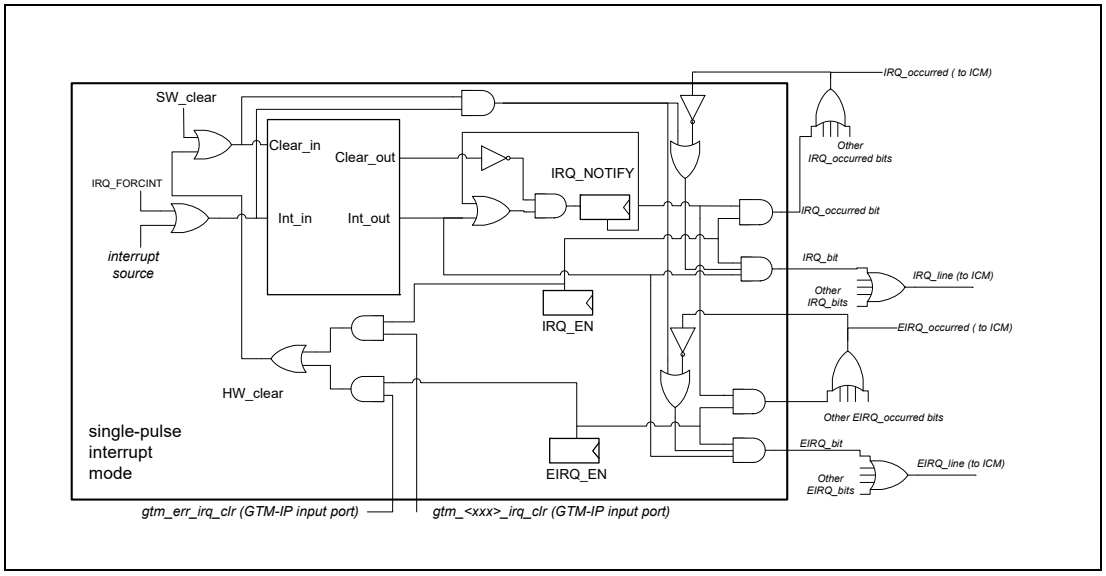



**Figure 25.16 Single-pulse interrupt scheme for modules AEI-bridge, TIM, MCS, CMP**

In Single-pulse Interrupt Mode, an error interrupt event is always captured in the register IRQ_NOTIFY, independent of the state of EIRQ_EN. However, only the first error interrupt event of an enabled error interrupt within a common error interrupt set is forwarded to signal EIRQ_line. Additional error interrupt events of the same error interrupt set cannot generate pulses on the signal EIRQ_line, until the corresponding bits in register IRQ_NOTIFY of enabled error interrupts are cleared by a clear event. The EIRQ_occurred signal line will be high, if the EIRQ_EN and the IRQ_NOTIFY register bits are set. The Single-pulse interrupt mode for error interrupts is shown in **Figure 25.16**.

To avoid unexpected EIRQ behavior in the single pulse mode, all desired error interrupt sources should be enabled by a single write access to EIRQ_EN and the notification bits should be cleared by a single write access to the register IRQ_NOTIFY.

An exception is the module ARU.

In this module the EIRQ_occurred bit of each error interrupt is directly connected (without OR-conjunction of neighboring EIRQ_occurred bits) to the inverter for suppressing further error interrupt pulses.

### 25.6.5.5 GTM-IP Interrupt concentration method

Because of the grouping of interrupts inside the ICM, it can be necessary for the software to access the ICM submodule first to determine the interrupt set that is responsible for an interrupt. A second access to the responsible register IRQ_NOTIFY is then necessary to identify the interrupt source, serve it and to reset the interrupt flag in register IRQ_NOTIFY afterwards. The interrupt flags are never reset by an access to the ICM. For a detailed description of the ICM submodule see **Section 25.15, Interrupt Concentrator Module (ICM)** .

## 25.6.6 GTM-IP Software Debugger Support

For software debugger support the GTM-IP comes with several features. E.g. status register bits must not be altered by a read access from a software debugger. To avoid this behavior to reset a status register bit by software, the CPU has to write a '1' explicitly to the register bit to reset its content.

The **Table 25.13** describes the behavior of some GTM-IP registers with special functionality on behalf of read accesses from the AEI bus interface.

### 25.6.6.1 Register behavior in case of Software Debugger accesses

**Table 25.13 Register behavior in case of Software Debugger accesses**

| Module | Register | Description |
|---|---|---|
| TIM | GTM0TIMixGPR0/1 | The overflow bit is not altered in case of a Debugger read access to this registers. |
| ATOM | GTM0ATOMixSR0/1 | In SOMC mode a read access to this register by the Debugger does not release the channel for a new compare/match event. |

Further on, some important states inside the GTM-IP submodule have to be signalled to the outside world, when reached and should for example trigger the software debugger to stop program execution. For this internal state signalling see the GTM-IP module integration guide.

The GTM provides an external signal gtm_halt, which disables clock signal SYS_CLK for debugging purposes. If SYS_CLK is disabled, a connected debugger can read any GTM related register and the GTM internal RAMs using AEI. Moreover, the debugger can also perform write accesses to the internal RAMs and to all GTM related registers in order to enable advanced debugging features (e.g. modifications of register contents in single step mode).

## 25.6.7 GTM-IP Programming conventions

To serve different application domains the GTM-IP is a highly configurable module with many configuration modes. In principle the submodules of the GTM-IP are intended to be configured at system start-up to fulfil certain functionality for the application domain the microcontroller runs in.

For example, a TIM input channel can be used to monitor an application specific external signal, and this signal has to be filtered. Therefore, the configuration of the TIM channel filter mode will be specific to the external signal characteristic. While it can be necessary to adapt the filter thresholds during runtime an adaptation of the filter mode during runtime is not reasonable. Thus, the change of the filter mode during runtime can lead to an unexpected behavior.

In general, the programmer has to be careful when reprogramming configuration registers of the GTM-IP submodules during runtime. It is recommended to disable the channels before reconfiguration takes place to avoid unexpected behavior of the GTM-IP.

### 25.6.8 GTM-IP TOP-Level Configuration Registers Overview

GTM-IP TOP-level contains following configuration registers:

**Table 25.14 Register list**

| Symbol | Register Name | Details in Section |
|---|---|---|
| GTM0GTMREV | GTM-IP Version control register | **25.6.9.1** |
| GTM0GTMRST | GTM-IP Global reset register | **25.6.9.2** |
| GTM0GTMCTRL | GTM-IP Global control register | **25.6.9.3** |
| GTM0GTMAEIADDRXPT | GTM-IP AEI Timeout exception address register | **25.6.9.4** |
| GTM0GTMIRQNOTIFY | GTM-IP Interrupt notification register | **25.6.9.5** |
| GTM0GTMIRQEN | GTM-IP Interrupt enable register | **25.6.9.6** |
| GTM0GTMEIRQEN | GTM-IP Error interrupt enable register | **25.6.9.12** |
| GTM0GTMIRQFORCINT | GTM-IP Software interrupt generation register | **25.6.9.7** |
| GTM0GTMIRQMODE | GTM-IP top level interrupts mode selection. Please note that this mode selection is only valid for the three interrupts described in **Section 25.6.9.5, GTM0GTMIRQNOTIFY** | **25.6.9.8** |
| GTM0GTMTIMiAUXINSRC (i = 0, 1) | GTM-IP TIM[i] module AUX_IN source selection register | **25.6.9.13** |
| GTM0GTMHWCONF | GTM-IP Hardware Configuration | **25.6.9.14** |
| GTM0GTMATOMiOUT (i = 0, 2) | GTM-IP ATOM output level | **25.6.9.15** |
| GTM0GTMEXTCAPENi (i = 0, 1) | GTM-IP trigger event forwarding in from TIM[i] and TIM[i+1] to MCS instance [i]. | **25.6.9.16** |

## 25.6.9 GTM-IP TOP-Level Configuration Registers Description

### 25.6.9.1 GTM0GTMREV

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00000_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEV_CODE2 | | | | DEV_CODE1 | | | | DEV_CODE0 | | | | MAJOR | | | |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MINOR | | | | NO | | | | STEP | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.15 GTM0GTMREV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | DEV_CODE2 | Device encoding digit 2. |
| 27 to 24 | DEV_CODE1 | Device encoding digit 1. |
| 23 to 20 | DEV_CODE0 | Device encoding digit 0. |
| 19 to 16 | MAJOR | Major version number<br>Define major version number of GTM-IP specification. |
| 15 to 12 | MINOR | Minor version number<br>Define minor version number of GTM-IP specification. |
| 11 to 8 | NO | Delivery number<br>Define delivery number of GTM-IP specification. |
| 7 to 0 | STEP | Release step<br>GTM Release step. |

**NOTES**

1. The numbers are encoded in BCD. Values “A” – “F” are characters.
2. See **Section 25.18.3.1, GTM0GTMREV** and **Section 25.19.3.1, GTM0GTMREV** for reset value.

### 25.6.9.2 GTM0GTMRST

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00004_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 25.16 GTM0GTMRST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 7 to 0 | STEP | GTM-IP Reset<br>0: No reset action<br>1: Initiate reset action for all submodules<br>**NOTES**<br>1. This bit is automatically cleared by hardware after it was written. Therefore, the register is always read as zero (0) by the software.<br>2. This bit is write protected by bit RF_PROT of **Section 25.6.9.3, GTM0GTMCTRL** |

### 25.6.9.3 GTM0GTMCTRL

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00008_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TO_VAL | | | | | — | — | TO_MODE | RF_PROT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W |

**Table 25.17 GTM0GTMCTRL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 8 to 4 | TO_VAL | AEI timeout value.<br>These bits define the number of cycles after which a timeout event occurs. When TO_VAL equals zero (0) the AEI timeout functionality is disabled. |
| 3, 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1 | TO_MODE | AEI timeout mode.<br>0: Observe: If timeout_counter=0 the address and rw signal in addition with timeout flag will be stored to the GTM0GTMAEIADDRXPT register. Following timeout_counter=0 accesses will not overwrite the first entry in the aei_addr_timeout register. Clearing the timeout flag/aei_status error_code will reenable the storing of a next faulty access.<br>1: Abort: In addition to observe mode the pending access will be aborted by signalling an illegal module access on aei_status and sending ready. In case of a read deliver as data 0 by serving of next AEI accesses. |
| 0 | RF_PROT | RST and FORCINT protection.<br>0: SW RST (global), SW interrupt FORCINT, and SW RAM reset functionality is enabled<br>1: SW RST (global), SW interrupt FORCINT, and SW RAM reset functionality is disabled |

### 25.6.9.4 GTM0GTMAEIADDRXPT

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $0000C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | TO_W1R0 | TO_ADDR | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TO_ADDR | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.18 GTM0GTMAEIADDRXPT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | These bits are always read as 0 |
| 20 | TO_W1R0 | AEI timeout Read/Write flag.<br>This bit defines the AEI Read/Write flag for which the AEI timeout event occurred. |
| 19 to 0 | TO_ADDR | AEI timeout address.<br>This bit field defines the AEI address for which the AEI timeout event occurred. |

### 25.6.9.5 GTM0GTMIRQNOTIFY

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00010_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | AEI_USP_BE | AEI_IM_ADDR | AEI_USP_ADDR | AEI_TO_XPT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 25.19 GTM0GTMIRQNOTIFY Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 3 | AEI_USP_BE | AEI unsupported byte enable interrupt.<br>0: No interrupt occurred<br>1: AEI_USP_BE interrupt was raised by the AEI interface<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 2 | AEI_IM_ADDR | AEI illegal Module address interrupt.<br>0: No interrupt occurred<br>1: AEI_IM_ADDR interrupt was raised by the AEI interface<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 1 | AEI_USP_ADDR | AEI unsupported address interrupt.<br>0: No interrupt occurred<br>1: AEI_USP_ADDR interrupt was raised by the AEI interface<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 0 | AEI_TO_XPT | AEI timeout exception occurred.<br>0: No interrupt occurred<br>1: AEI_TO_XPT interrupt was raised by the AEI Timeout detection unit<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

### 25.6.9.6 GTM0GTMIRQEN

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00014_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | AEI_USP_BE_IRQ_EN | AEI_IM_ADDR_IRQ_EN | AEI_USP_ADDR_IRQ_EN | AEI_TO_XPT_IRQ_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 25.20 GTM0GTMIRQEN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 3 | AEI_USP_BE_IRQ_EN | AEI_USP_BE_IRQ interrupt enable.<br>0: Disable interrupt, interrupt is not visible outside GTM-IP<br>1: Enable interrupt, interrupt is visible outside GTM-IP |
| 2 | AEI_IM_ADDR_IRQ_EN | AEI_IM_ADDR_IRQ interrupt enable.<br>0: Disable interrupt, interrupt is not visible outside GTM-IP<br>1: Enable interrupt, interrupt is visible outside GTM-IP |
| 1 | AEI_USP_ADDR_IRQ_EN | AEI_USP_ADDR_IRQ interrupt enable.<br>0: Disable interrupt, interrupt is not visible outside GTM-IP<br>1: Enable interrupt, interrupt is visible outside GTM-IP |
| 0 | AEI_TO_XPT_IRQ_EN | AEI_TO_XPT_IRQ interrupt enable.<br>0: Disable interrupt, interrupt is not visible outside GTM-IP<br>1: Enable interrupt, interrupt is visible outside GTM-IP |

### 25.6.9.7 GTM0GTMIRQFORCINT

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00018_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TRG_AEI_USP_BE | TRG_AEI_IM_ADDR | TRG_AEI_USP_ADDR | TRG_AEI_TO_XPT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 25.21 GTM0GTMIRQFORCINT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 3 | TRG_AEI_USP_BE | Trigger AEI_USP_BE_IRQ interrupt by software.<br>0: No interrupt triggering<br>1: Assert AEI_USP_BE_IRQ interrupt for one clock cycle<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of **Section 25.6.9.3, GTM0GTMCTRL** |
| 2 | TRG_AEI_IM_ADDR | Trigger AEI_IM_ADDR_IRQ interrupt by software.<br>0: No interrupt triggering<br>1: Assert AEI_IM_ADDR_IRQ interrupt for one clock cycle<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of **Section 25.6.9.3, GTM0GTMCTRL** |
| 1 | TRG_AEI_USP_ADDR | Trigger AEI_USP_ADDR_IRQ interrupt by software.<br>0: No interrupt triggering<br>1: Assert AEI_USP_ADDR_IRQ interrupt for one clock cycle<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of **Section 25.6.9.3, GTM0GTMCTRL** |
| 0 | TRG_AEI_TO_XPT | Trigger AEI_TO_XPT_IRQ interrupt by software.<br>0: No interrupt triggering<br>1: Assert AEI_TO_XPT_IRQ interrupt for one clock cycle<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of **Section 25.6.9.3, GTM0GTMCTRL** |

### 25.6.9.8 GTM0GTMIRQMODE

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $0001C_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IRQ_MODE | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 25.22 GTM0GTMIRQMODE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 0 | IRQ_MODE | Interrupt strategy mode selection for the AEI timeout and address monitoring interrupts.<br>00: Level mode<br>01: Pulse mode<br>10: Pulse-Notify mode<br>11: Single-Pulse mode<br>The interrupt modes are described in **Section 25.6.5, GTM-IP Interrupt Concept**. |

### 25.6.9.9 GTM0GTMBRIDGEMODE

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00030_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BUFF_DPT | | | | | | | | — | — | — | — | — | — | — | BRG_RST |
| Value after reset | — | — | — | — | — | — | — | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | SYNC_INPUT_REG | — | — | BUFF_OVL | MODE_UP_PGR | — | — | — | — | — | — | MSK_WR_RSP | BRG_MODE |
| Value after reset | 0 | 0 | 0 | — | 0 | 0 | — | — | 0 | 0 | 0 | 0 | 0 | 0 | — | — |
| R/W | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R/W | R/W |

**Table 25.23 GTM0GTMBRIDGEMODE Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | BUFF_DPT | Buffer depth of AEI bridge.<br>Signals the buffer depth of the GTM AEI bridge implementation.<br>**NOTE**<br>Reset value depends on the hardware configuration chosen by silicon vendor. For detail, see **Section 25.18.4.1, GTM0GTMBRIDGEMODE**, or **Section 25.19.4.1, GTM0GTMBRIDGEMODE**. |
| 23 to 17 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 16 | BRG_RST | Bridge software reset.<br>0: No bridge reset request.<br>1: Bridge reset request.<br>**NOTE**<br>This bit is cleared automatically after write. |
| 15 to 13 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 12 | SYNC_INPUT_REG | Additional pipelined stage in synchronous bridge mode<br>0: No additional pipelined stage implemented.<br>1: Additional pipelined stage implemented. All accesses in synchronous mode will be increased by one clock cycle.<br>**NOTE**<br>Reset value depends on the hardware configuration chosen by silicon vendor. For detail, see **Section 25.18.4.1, GTM0GTMBRIDGEMODE**, or **Section 25.19.4.1, GTM0GTMBRIDGEMODE**. |
| 11, 10 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 9 | BUFF_OVL | Buffer overflow register.<br>0: No buffer overflow occurred.<br>1: Buffer overflow occurred.<br>A buffer overflow can occur while multiple aborts are issued by the external bus or a pipelined instruction is started while FBC = 0 (see GTM0GTMBRIDGEPTR1 register). |
| 8 | MODE_UP_PGR | Mode update in progress.<br>0: No update in progress.<br>1: Update in progress. |

**Table 25.23 GTM0GTMBRIDGEMODE Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1 | MSK_WR_RSP | Mask write response.<br>0: Do not mask the write response<br>1: Mask write response |
| 0 | BRG_MODE | Defines the operation mode for the AEI bridge.<br>0: AEI bridge operates in sync_bridge mode<br>1: AEI bridge operates in async_bridge mode<br>**NOTE**<br>Reset value depends on the hardware configuration chosen by silicon vendor. For detail, see **Section 25.18.4.1, GTM0GTMBRIDGEMODE**, or **Section 25.19.4.1, GTM0GTMBRIDGEMODE**. |

### 25.6.9.10 GTM0GTMBRIDGEPTR1

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00034_H$

**Value after reset:** $0xx0\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSP_TRAN_RDY | | | | | | FBC | | | | | | ABT_TRAN_PGR | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | – | – | – | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ABT_TRAN_PGR | TRAN_IN_PGR | | | | | FIRST_RSP_PTR | | | | | NEW_TRAN_PTR | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.24 GTM0GTMBRIDGEPTR1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | RSP_TRAN_RDY | Response transactions ready.<br>Amount of ready response transactions. |
| 25 to 20 | FBC | Free buffer count.<br>Number of free buffer entries.<br>**NOTE**<br>Initial value depends on the hardware configuration chosen by silicon vendor. (see BUFF_DPT in GTM0GTMBRIDGEMODE register). |
| 19 to 15 | ABT_TRAN_PGR | Aborted transaction in progress pointer. |
| 14 to 10 | TRAN_IN_PGR | Transaction in progress pointer (acquire) |
| 9 to 5 | FIRST_RSP_PTR | First response pointer.<br>Signals the actual value of first response pointer. |
| 4 to 0 | NEW_TRAN_PTR | New transaction pointer.<br>Signals the actual value of the new transaction pointer. |

**NOTES**

1. This register operates on the AEI_CLK domain.
2. This register holds diagnosis information about the AEI bus bridge. Each access to the GTM_IP will update the defined pointer bit fields. Depending on the mode of GTM_MODE_BRIDGE (BRG_MODE, MSK_WR_RESP), the AEI protocol and operating frequency which is use, the 4 pointer bitfields will change at different clock cycles relative to the start of the transaction. This leads to the fact that reading the register can show values not equal to the defined Initial Value, even directly after a write to GTM0GTMBRIDGEMODE with BRG_RST = 1 was done.

### 25.6.9.11 GTM0GTMBRIDGEPTR2

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00038_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | TRAN_IN_PGR2 | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.25 GTM0GTMBRIDGEPTR2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | These bits are always read as 0. |
| 4 to 0 | TRAN_IN_ PGR2 | Transaction in progress pointer (aquire2) |

**NOTE**

This register operates on the GTM_CLK domain.

### 25.6.9.12 GTM0GTMEIRQEN

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00020_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | AEI_USP_BE_EIRQ_EN | AEI_IM_ADDR_EIRQ_EN | AEI_USP_ADDR_EIRQ_EN | AEI_TO_XPT_EIRQ_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 25.26 GTM0GTMEIRQEN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 3 | AEI_USP_BE_EIRQ_EN | AEI_USP_BE_EIRQ error interrupt enable.<br>0: Disable error interrupt, interrupt is not visible outside GTM-IP<br>1: Enable error interrupt, interrupt is visible outside GTM-IP |
| 2 | AEI_IM_ADDR_EIRQ_EN | AEI_IM_ADDR_EIRQ error interrupt enable.<br>0: Disable error interrupt, interrupt is not visible outside GTM-IP<br>1: Enable error interrupt, interrupt is visible outside GTM-IP |
| 1 | AEI_USP_ADDR_EIRQ_EN | AEI_USP_ADDR_EIRQ error interrupt enable.<br>0: Disable error interrupt, interrupt is not visible outside GTM-IP<br>1: Enable error interrupt, interrupt is visible outside GTM-IP |
| 0 | AEI_TO_XPT_EIRQ_EN | AEI_TO_XPT_EIRQ error interrupt enable.<br>0: Disable error interrupt, interrupt is not visible outside GTM-IP<br>1: Enable error interrupt, interrupt is visible outside GTM-IP |

### 25.6.9.13 GTM0GTMTIMiAUXINSRC (i = 0, 1)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0GTMTIM0AUXINSRC: <GTM_base> + $00040_H$
GTM0GTMTIM1AUXINSRC: <GTM_base> + $00044_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | SRC_CH7 | SRC_CH6 | SRC_CH5 | SRC_CH4 | SRC_CH3 | SRC_CH2 | SRC_CH1 | SRC_CH0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.27 GTM0GTMTIMiAUXINSRC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 7 | SRC_CH7 | Defines AUX_IN source of TIM[i] channel 7<br>0: TOM Output selected TOM[a] channel [b]<br>with a = (i* 8 + 7) div 16; b = (i* 8 + 7) mod 16;<br>1: ATOM Output selected ATOM[i] channel[x] |
| 6 | SRC_CH6 | Defines AUX_IN source of TIM[i] channel 6<br>0: TOM Output selected TOM[a] channel [b]<br>with a = (i* 8 + 6) div 16; b = (i* 8 + 6) mod 16;<br>1: ATOM Output selected ATOM[i] channel[x] |
| 5 | SRC_CH5 | Defines AUX_IN source of TIM[i] channel 5<br>0: TOM Output selected TOM[a] channel [b]<br>with a = (i* 8 + 5) div 16; b = (i* 8 + 5) mod 16;<br>1: ATOM Output selected ATOM[i] channel[x] |
| 4 | SRC_CH4 | Defines AUX_IN source of TIM[i] channel 4<br>0: TOM Output selected TOM[a] channel [b]<br>with a = (i* 8 + 4) div 16; b = (i* 8 + 4) mod 16;<br>1: ATOM Output selected ATOM[i] channel[x] |
| 3 | SRC_CH3 | Defines AUX_IN source of TIM[i] channel 3<br>0: TOM Output selected TOM[a] channel [b]<br>with a = (i* 8 + 3) div 16; b = (i* 8 + 3) mod 16;<br>1: ATOM Output selected ATOM[i] channel[x] |
| 2 | SRC_CH2 | Defines AUX_IN source of TIM[i] channel 2<br>0: TOM Output selected TOM[a] channel [b]<br>with a = (i* 8 + 2) div 16; b = (i* 8 + 2) mod 16;<br>1: ATOM Output selected ATOM[i] channel[x] |
| 1 | SRC_CH1 | Defines AUX_IN source of TIM[i] channel 1<br>0: TOM Output selected TOM[a] channel [b]<br>with a = (i* 8 + 1) div 16; b = (i* 8 + 1) mod 16;<br>1: ATOM Output selected ATOM[i] channel[x] |
| 0 | SRC_CH0 | Defines AUX_IN source of TIM[i] channel 0<br>0: TOM Output selected TOM[a] channel [b]<br>with a = (i* 8) div 16; b = (i* 8) mod 16;<br>1: ATOM Output selected ATOM[i] channel[x] |

### 25.6.9.14 GTM0GTMHWCONF

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00024_H$

**Value after reset:** $000X\ XX0X_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | IRQ_MODE_SINGLE_PULSE | IRQ_MODE_PULSE_NOTIFY | IRQ_MODE_PULSE | IRQ_MODE_LEVEL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | ARU_CONNECT_CONFIG | ERM | RAM_INIT_RST | — | — | — | — | ATOM_TRIG_CHAIN | | | ATOM_OUT_RST | — | SYNC_INPUT_REG | BRIDGE_MODE_RST | GRSTEN |
| Value after reset | 0 | — | — | — | 0 | 0 | 0 | 0 | — | — | — | — | 0 | 0 | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.28 GTM0GTMHWCONF Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | These bits are always read as 0. |
| 19 | IRQ_MODE_SINGLE_PULSE | IRQ_MODE_SINGLE_PULSE<br>0: single pulse mode not available<br>1: single pulse mode available |
| 18 | IRQ_MODE_PULSE_NOTIFY | IRQ_MODE_PULSE_NOTIFY<br>0: pulse notify mode not available<br>1: pulse notify mode available |
| 17 | IRQ_MODE_PULSE | IRQ_MODE_PULSE<br>0: pulse mode not available<br>1: pulse mode available |
| 16 | IRQ_MODE_LEVEL | IRQ_MODE_LEVEL<br>0: level mode not available<br>1: level mode available |
| 15 | Reserved | These bits are always read as 0. |
| 14 | ARU_CONNECT_CONFIG | Defines number of parallel ARU ports<br>0: 2 ARU ports available (two independent counter)<br>1: 1 ARU port available |
| 13 | ERM | ERM: enable RAM1 MSB for available MCS modules<br>0: MSB of RAM1 address not used<br>1: MSB of RAM1 address used |
| 12 | RAM_INIT_RST | RAM_INIT_RST: RAM initialization from reset<br>0: RAM is not initialized after reset<br>1: RAM is initialized after reset |
| 11 to 8 | Reserved | These bits are always read as 0. |
| 7 to 5 | ATOM_TRIG_CHAIN | ATOM trigger chain length without synchronization register<br><br>It defines after which ATOM instance count a synchronization register is introduced into trigger chain (after ATOM_TRIG_<i> output if instance i and ATOM_TRIG_<i+1> input of instance i+1).<br>Valid values are 1 to 7. 1 means that after each instance a synchronization register is placed. |

**Table 25.28 GTM0GTMHWCONF Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 | ATOM_OUT_RST | ATOM_OUT reset level<br>0: ATOM_OUT reset level is '0'<br>1: ATOM_OUT reset level is '1'<br>**NOTE**<br>This reset level defines the reset value of bit SL for all ATOM channel |
| 3 | Reserved | These bits are always read as 0. |
| 2 | SYNC_INPUT_REG | Additional pipelined stage in synchronous bridge mode<br>0: No additional pipelined stage implemented.<br>1: Additional pipelined stage implemented. All accesses in synchronous mode will be increased by one clock cycle. |
| 1 | BRIDGE_MODE_RST | Bridge mode after reset<br>0: Bridge starts in synchronous mode after reset<br>1: Bridge starts in asynchronous mode after reset |
| 0 | GRSTEN | Global Reset Enable<br>0: Global GTM reset register disabled<br>1: Global GTM reset register enabled |

### 25.6.9.15 GTM0GTMATOMiOUT (i = 0, 2)

**Access:** This register can be read in 32-bit units.

**Address:** GTM0GTMATOM0OUT: <GTM_base> + $00098_H$
GTM0GTMATOM2OUT: <GTM_base> + $0009C_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ATOM_IP1_OUT_N | | | | | | | | ATOM_IP1_OUT | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ATOM_I_OUT_N | | | | | | | | ATOM_I_OUT | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.29 GTM0GTMATOMiOUT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | ATOM_IP1_OUT_N | Output level snapshot of ATOM[i+1]_OUT_N all channels<br>Actual level of primary output ports ATOM[i+1]_OUT_N of channel 0 to 7 (after DTM)<br>**NOTE**<br>Reset value depends on the hardware configuration chosen by silicon vendor. See **Section 25.6.9.14, GTM0GTMHWCONF** for chosen value. |
| 23 to 16 | ATOM_IP1_OUT | Output level snapshot of ATOM[i+1]_OUT all channels<br>Actual level of primary output ports ATOM[i+1]_OUT of channel 0 to 7 (after DTM)<br>**NOTE**<br>Reset value depends on the hardware configuration chosen by silicon vendor. See **Section 25.6.9.14, GTM0GTMHWCONF** for chosen value. |
| 15 to 8 | ATOM_I_OUT_N | Output level snapshot of ATOM[i]_OUT_N all channels<br>Actual level of primary output ports ATOM[i]_OUT_N of channel 0 to 7 (after DTM)<br>**NOTE**<br>Reset value depends on the hardware configuration chosen by silicon vendor. See **Section 25.6.9.14, GTM0GTMHWCONF** for chosen value. |
| 7 to 0 | ATOM_I_OUT | Output level snapshot of ATOM[i]_OUT all channels<br>Actual level of primary output ports ATOM[i]_OUT of channel 0 to 7 (after DTM)<br>**NOTE**<br>Reset value depends on the hardware configuration chosen by silicon vendor. See **Section 25.6.9.14, GTM0GTMHWCONF** for chosen value. |

**NOTE**

Reset value depends on the hardware configuration chosen by silicon vendor. See **Section 25.6.9.14, GTM0GTMHWCONF** for chosen value.

### 25.6.9.16 GTM0GTMEXTCAPENi (i = 0, 1)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0GTMEXTCAPEN0: <GTM_base> + $005C_H$
GTM0GTMEXTCAPEN1: <GTM_base> + $0060_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIM_IP1_EXT_CAP_EN | | | | | | | | TIM_I_EXT_CAP_EN | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.30 GTM0GTMEXTCAPENi Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 8 | TIM_IP1_EXT_CAP_EN | TIM_IP1_EXT_CAP_EN: TIM [i+1]_EXT_CAPTURE signal forwarding enable<br>0: Disable forwarding of signal TIM [i+1]_EXT_CAPTURE to MCS[i]<br>1: Enable forwarding of signal TIM [i+1]_EXT_CAPTURE to MCS[i] |
| 7 to 0 | TIM_I_EXT_CAP_EN | TIM [i]_EXT_CAPTURE signal forwarding enable<br>0: Disable forwarding of signal TIM [i]_EXT_CAPTURE to MCS [i]<br>1: Enable forwarding of signal TIM [i]_EXT_CAPTURE to MCS [i] |

## 25.7 Advanced Routing Unit (ARU)

### 25.7.1 Overview

The Advanced Routing Unit (ARU) is a flexible infrastructure component for transferring 53 bit wide data (five control bits and two 24 bit values) between several submodules of the GTM core in a configurable manner.

Since the concept of the ARU has already been described in **Section 25.6.3, ARU Routing Concept**, this section only describes additional ARU features that can be used by the software for configuring and debugging ARU related data streams.

Also the definition of ‘streams’ and ‘channels’ in the ARU context is done in **Section 25.6.3, ARU Routing Concept**.

### 25.7.2 Special Data Sources

Besides the addresses of the submodule related data sources as described in **Table 25.204, ARU Write Address Overview**, **Table 25.211, ARU Write Address Overview**, the ARU provides two special data sources that can be used for the configuration of data streams. These data sources are defined as follows:

Address $1FF_H$: Data source that provides always a 53 bit data word with zeros. A read access to this memory location will never block a requesting data destination.

Address $1FE_H$: Data source that never provides a data word. A read access to this memory location will always block a requesting data destination. This is the reset value of the read registers inside the data destinations.

Address $000_H$: This address is reserved and can be used to bring data through the ARU registers GTM0ARUDATAH and GTM0ARUDATAL into the system by writing the write address $000_H$ into the GTM0ARUACCESS register. This means that software test data can be brought into the GTM-IP by the CPU.

### 25.7.3 ARU Access via AEI

Besides the data transfer between the connected submodules, there are two possibilities to access ARU data via the AEI.

#### 25.7.3.1 Default ARU Access

The default ARU access incorporates the registers GTM0ARUACCESS, which is used for initiation of a read or write request and the registers GTM0ARUDATAH and GTM0ARUDATAL that provide the ARU data word to be transferred.

The status of a read or write transfer can be determined by polling specific bits in register GTM0ARUACCESS. Furthermore the acc_ack bit in the interrupt notify register is set after the read or write access is performed to avoid data loss e.g. on access cancelation.

A pending read or write request may also be cancelled by clearing the associated bit.

In the case of a read request, the AEI access behaves as a read request initiated by a data destination of a module. The read request is served by the ARU immediately when no other destination has a pending read request. This means, that an AEI read access does not take part in the scheduling of the destination channels and that the time between two consecutive read accesses is not limited by the round trip time.

On the other hand, the AEI access has the lowest priority behind the ARU scheduler that serves the destination channels. Thus, in worst case, the read request is served after one round trip of the ARU, when all destination channels would request data at the same point in time.

In the case of the write request, the ARU provides the write data at the address defined by the ADDR bit field inside the GTM0ARUACCESS register.

To avoid data loss, the reserved ARU address $0_H$ has to be used to bring data into the system. Otherwise, in case the address specified inside the ADDR bit field is defined for another submodule that acts as a source at the ARU data loss may occur and no deterministic behavior is guaranteed.

This is because the regular source submodule is not aware that its address is used by the ARU itself to provide data to a destination.

It is guaranteed that the ARU write data is send to the destination in case of both modules want to provide data at the same time.

Configuring both read and write request bits results in a read request, if the write request bit inside the register isn't already set. The read request bit will be set but not the write request bit. The following table describes the important cases of the bit 12 (RREQ) and bit 13 (WREQ) of the GTM0ARUACCESS register:

**Table 25.31 ARU Access**

| AEI write access : aei_wdata (13:12) | Actual value of GTM0ARUACCESS (13:12) | Next value of GTM0ARUACCESS (13:12) | Comment |
|---|---|---|---|
| 0 0 | 0 1 | 0 0 | Cancel read request |
| 0 0 | 1 0 | 0 0 | Cancel write request |
| 0 1 | 1 0 | 1 0 | Unchanged register |
| 1 0 | 0 1 | 0 1 | Unchanged register |
| 1 1 | 0 0 | 0 1 | Both read and write request results in a read request |
| 1 1 | 1 0 | 1 0 | As before but WREQ bit is already set → unchanged register |

### 25.7.3.2 Debug Access

The debug access mode enables to inspect routed data of configured data streams during runtime.

The ARU provides two independent debug channels, whereas each is configured by a dedicated ARU read address in register GTM0ARUDBGACCESS0 and GTM0ARUDBGACCESS1 respectively.

The registers GTM0ARUDBGDATA0H and GTM0ARUDBGDATA0L (GTM0ARUDBGDATA1H and GTM0ARUDBGDATA1L) provide read access to the latest data word that the corresponding data source sent through the ARU.

Any time when data is transferred through the ARU from a data source to the destination requesting the data the interrupt signal ARU_NEW_DATA0_IRQ (ARU_NEW_DATA1_IRQ) is raised.

For advanced debugging purposes, the interrupt signal can also be triggered by software using the register GTM0ARUIRQFORCINT.

Please note, that the debug mechanism should not be used by the application, when a HW-Debugger is used to trace the ARU communication. In that case, the debug registers are used by the HW-Debugger to specify the ARU streams that should be traced.

### 25.7.4 ARU Interrupt Signals

The following table describes ARU interrupt signals:

**Table 25.32 ARU Interrupt Signals**

| Signal | Description |
|---|---|
| ARU_NEW_DATA0_IRQ | Indicates that data is transferred through the ARU using debug channel GTM0ARUDBGACCESS0. |
| ARU_NEW_DATA1_IRQ | Indicates that data is transferred through the ARU using debug channel GTM0ARUDBGACCESS1. |
| ACC_ACK_IRQ | ARU access acknowledge IRQ. |

### 25.7.5 ARU Configuration Registers Overview

The following table shows a conclusion of configuration registers address offsets and initial values.

**Table 25.33 Register List**

| Symbol | Register Name | Details in Section |
|---|---|---|
| GTM0ARUACCESS | ARU access register | **25.7.6.1** |
| GTM0ARUDATAH | ARU access register upper data word | **25.7.6.2** |
| GTM0ARUDATAL | ARU access register lower data word | **25.7.6.3** |
| GTM0ARUDBGACCESS0 | Debug access channel 0 | **25.7.6.4** |
| GTM0ARUDBGDATA0H | Debug access 0 transfer register upper data word | **25.7.6.5** |
| GTM0ARUDBGDATA0L | Debug access 0 transfer register lower data word | **25.7.6.6** |
| GTM0ARUDBGACCESS1 | Debug access channel 0 | **25.7.6.7** |
| GTM0ARUDBGDATA1H | Debug access 1 transfer register upper data word | **25.7.6.8** |
| GTM0ARUDBGDATA1L | Debug access 1 transfer register lower data word | **25.7.6.9** |
| GTM0ARUIRQNOTIFY | ARU Interrupt notification register | **25.7.6.10** |
| GTM0ARUIRQEN | ARU Interrupt enable register | **25.7.6.11** |
| GTM0ARUIRQFORCINT | Register for forcing the ARU_NEW_DATA_IRQ interrupt | **25.7.6.12** |
| GTM0ARUIRQMODE | IRQ mode configuration register | **25.7.6.13** |
| GTM0ARUCADDREND | ARU caddr counter end value | **25.7.6.14** |

## 25.7.6 ARU Configuration Registers Description

### 25.7.6.1 GTM0ARUACCESS

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00280_H$

**Value after reset:** 0000 $01FE_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | WREQ | RREQ | — | — | — | ADDR | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| R/W | R | R | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.34 GTM0ARUACCESS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 13 | WREQ | Initiate write request<br>0: No write request is pending<br>1: Mark data in registers GTM0ARUDATAH and GTM0ARUDATAL as valid<br>**NOTES**<br>1. This bit is cleared automatically after transaction. Moreover, it can be cleared by software to cancel a write request.<br>2. WREQ bit are only writable if RREQ bit is zero, so to switch from WREQ to RREQ a cancel request has to be performed before.<br>3. Configuring both RREQ and WREQ bits results in a read request, so WREQ bit will not be set<br>4. The data is provided at address ADDR. This address has to be programmed as the source address in the destination submodule channel. In worst case, the data is provided after one full ARU round trip. |
| 12 | RREQ | Initiate read request<br>0: No read request is pending<br>1: Set read request to source channel addressed by ADDR<br>**NOTES**<br>1. This bit is cleared automatically after transaction. Moreover, it can be cleared by software to cancel a read request.<br>2. RREQ bit are only writable if WREQ bit is zero, so to switch from RREQ to WREQ a cancel request has to be performed before.<br>3. Configuring both RREQ and WREQ bits results in a read request, so RREQ bit will be set if the WREQ bit of the register isn't already set.<br>4. The ARU read request on address ADDR is served immediately when no other destination has actually a read request when the RREQ bit is set by CPU. In a worst case scenario, the read request is served after one round trip of the ARU, but this is only the case when every destination channel issues a read request at consecutive points in time. |
| 11 to 9 | Reserved | These bits are always read as 0. When written, write the initial value. |

**Table 25.34 GTM0ARUACCESS Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 8 to 0 | ADDR | ARU address<br>Define the ARU address used for transferring data<br>**NOTES**<br>1. For an ARU write request, the preferred address $0_H$ have to be used.<br>2. A write request to the address $1FF_H$ (always full address) or $1FE_H$ (always empty address) are ignored and doesn't have any effect.<br>3. ARU address bits ADDR are only writable if RREQ and WREQ bits are zero |

**Note:** The register GTM0ARUACCESS can be used either for reading or for writing at the same point in time.

### 25.7.6.2 GTM0ARUDATAH

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00284_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DATA | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.35 GTM0ARUDATAH Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 28 to 0 | DATA | Upper ARU data word<br>Transfer upper ARU data word addressed by ADDR. The data bits 24 to 52 of an ARU word are mapped to the data bits 0 to 28 of this register |

### 25.7.6.3 GTM0ARUDATAL

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00288_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DATA | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.36 GTM0ARUDATAL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. |
| 28 to 0 | DATA | Lower ARU data word<br>**NOTES**<br>1. Transfer lower ARU data word addressed by ADDR. The data bits 0 to 23 of an ARU word are mapped to the data bits 0 to 23 of this register and the data bits 48 to 52 of an ARU word are mapped to the data bits 24 to 28 of this register when data is read by the CPU.<br>2. For writing data into the ARU by the CPU the bits 24 to 28 are not transferred to bit 48 to 52 of the ARU word. Only bits 0 to 23 are written to bits 0 to 23 of the ARU word. |

### 25.7.6.4 GTM0ARUDBGACCESS0

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $0028C_H$

**Value after reset:** $0000\ 01FE_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ADDR | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.37 GTM0ARUDBGACCESS0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 8 to 0 | ADDR | ARU debugging address<br>Define address of ARU debugging channel 0. |

### 25.7.6.5 GTM0ARUDBGDATA0H

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00290_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DATA | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.38 GTM0ARUDBGDATA0H Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. |
| 28 to 0 | DATA | Upper debug data word<br>**NOTES**<br>1. Transfer upper ARU data word addressed by register GTM0ARUDBGACCESS0. The data bits 24 to 52 of an ARU word are mapped to the data bits 0 to 28 of this register.<br>2. The interrupt ARU_NEW_DATA0_IRQ is raised if a new data word is available. |

### 25.7.6.6 GTM0ARUDBGDATA0L

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00294_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DATA | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.39 GTM0ARUDBGDATA0L Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. |
| 28 to 0 | DATA | Lower debug data word<br>**NOTES**<br>1. Transfer lower ARU data word addressed by register GTM0ARUDBGACCESS0. The data bits 0 to 23 of an ARU word are mapped to the data bits 0 to 23 of this register and the data bits 48 to 52 of an ARU word is mapped to the data bits 24 to 28 of this register.<br>2. The interrupt ARU_NEW_DATA0_IRQ is raised if a new data word is available. |

### 25.7.6.7 GTM0ARUDBGACCESS1

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00298_H$

**Value after reset:** $0000\ 01FE_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ADDR | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.40 GTM0ARUDBGACCESS1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 8 to 0 | ADDR | ARU debugging address<br>Define address of ARU debugging channel 1.<br>**NOTE**<br>Define address of ARU debugging channel 1. |

### 25.7.6.8 GTM0ARUDBGDATA1H

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $0029C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DATA | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.41 GTM0ARUDBGDATA1H Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. |
| 28 to 0 | DATA | Upper debug data word<br>**NOTES**<br>1. Transfer upper ARU data word addressed by register GTM0ARUDBGACCESS1. The data bits 24 to 52 of an ARU word are mapped to the data bits 0 to 28 of this register.<br>2. The interrupt ARU_NEW_DATA1_IRQ is raised if a new data word is available. |

### 25.7.6.9 GTM0ARUDBGDATA1L

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $002A0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DATA | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.42 GTM0ARUDBGDATA1L Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. |
| 28 to 0 | DATA | Lower debug data word<br>**NOTES**<br>1. Transfer lower ARU data word addressed by register GTM0ARUDBGACCESS1. The data bits 0 to 23 of an ARU word are mapped to the data bits 0 to 23 of this register and the data bits 48 to 52 of an ARU word is mapped to the data bits 24 to 28 of this register.<br>2. The interrupt ARU_NEW_DATA1_IRQ is raised if a new data word is available. |

### 25.7.6.10 GTM0ARUIRQNOTIFY

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $002A4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ACC_ACK | NEW_DATA1 | NEW_DATA0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 25.43 GTM0ARUIRQNOTIFY Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 | ACC_ACK | AEI to ARU access finished, on read access data are valid<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 1 | NEW_DATA1 | Data was transferred for addr GTM0ARUDBGACCESS1<br>0: No interrupt occurred<br>1: ARU_NEW_DATA1_IRQ interrupt was raised by the ARU<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 0 | NEW_DATA0 | Data was transferred for addr GTM0ARUDBGACCESS0<br>0: No interrupt occurred<br>1: ARU_NEW_DATA0_IRQ interrupt was raised by the ARU<br>**NOTE**<br>Reset value depends on the hardware configuration chosen by silicon vendor. See GTM0GTMHWCONF for chosen value. |

### 25.7.6.11 GTM0ARUIRQEN

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $002A8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ACC_ACK_IRQ | NEW_DATA1_IR | NEW_DATA0_IR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 25.44 GTM0ARUIRQEN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 | ACC_ACK_IRQ | ACC_ACK_IRQ interrupt enable<br>0: Disable interrupt, interrupt is not visible outside GTM-IP<br>1: Enable interrupt, interrupt is visible outside GTM-IP |
| 1 | NEW_DATA1_IR | ARU_NEW_DATA1_IRQ interrupt enable<br>0: Disable interrupt, interrupt is not visible outside GTM-IP<br>1: Enable interrupt, interrupt is visible outside GTM-IP |
| 0 | NEW_DATA0_IR | ARU_NEW_DATA0_IRQ interrupt enable<br>0: Disable interrupt, interrupt is not visible outside GTM-IP<br>1: Enable interrupt, interrupt is visible outside GTM-IP |

### 25.7.6.12 GTM0ARUIRQFORCINT

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $002AC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TRG_ACC_ACK | TRG_NEW_DAT1 | TRG_NEW_DAT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 25.45 GTM0ARUIRQFORCINT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 | TRG_ACC_ACK | Trigger ACC_ACK interrupt<br>0: Corresponding bit in status register will not be forced<br>1: Assert corresponding field in GTM0ARUIRQNOTIFY register<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of register GTM0GTMCTRL. |
| 1 | TRG_NEW_DAT1 | Trigger new data 1 interrupt<br>0: Corresponding bit in status register will not be forced<br>1: Assert corresponding field in GTM0ARUIRQNOTIFY register<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of register GTM0GTMCTRL. |
| 0 | TRG_NEW_DAT0 | Trigger new data 0 interrupt<br>0: Corresponding bit in status register will not be forced<br>1: Assert corresponding field in GTM0ARUIRQNOTIFY register<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of register GTM0GTMCTRL. |

### 25.7.6.13 GTM0ARUIRQMODE

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $002B0_H$

**Value after reset:** $0000\ 000X_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IRQ_MODE | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 25.46 GTM0ARUIRQMODE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1, 0 | IRQ_MODE | IRQ mode selection<br>00: Level mode<br>01: Pulse mode<br>10: Pulse-Notify mode<br>11: Single-Pulse mode<br>**NOTE**<br>The interrupt modes are described in **Section 25.6.5, GTM-IP Interrupt Concept** . |

### 25.7.6.14 GTM0ARUCADDREND

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $002B4_H$

**Value after reset:** $0000\ 00XX_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CADDR_END | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — | — | — | — | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.47 GTM0ARUCADDREND Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 6 to 0 | CADDR_END | Set end value of ARU caddr counter<br>**NOTES**<br>1. The ARU roundtrip counter aru_caddr runs from zero to caddr_end value.<br>2. Shorten the ARU roundtrip cycle by setting a smaller number than the defined reset value will cause that not all ARU-connected modules will be served.<br>3. Making the roundtrip cycle longer than the reset value would cause longer ARU roundtrip time and as a result some ARU-connected modules will not be served as fast as possible for this device.<br>4. The reset value is device-specific. For more information see **Section 25.18.3.2, GTM0ARUCADDREND** and **Section 25.19.3.2, GTM0ARUCADDREND**.<br>5. This bit is write protected by bit RF_PROT of register GTM0GTMCTRL. |

# 25.8 Clock Management Unit (CMU)

## 25.8.1 Overview

The Clock Management Unit (CMU) is responsible for clock generation of the counters and of the GTM-IP. The CMU generate different clock sources for the whole GTM-IP. **Figure 25.17** shows a block diagram of the CMU.

The Configurable Clock Generation (CFGU) subunit provides eight dedicated clock sources for the following GTM submodules: TIM, ATOM, TBU, and MON. Each instance of such a submodule can choose an arbitrary clock source, in order to specify wide-ranging time bases.

The clock source signals CMU_CLK[x] (x = 0 to 7) are implemented in form of enable signals for the corresponding registers, which means that the actual clock signal of all registers always use the SYS_CLK signal.

The four configurable clock signals CMU_CLK0, CMU_CLK1, CMU_CLK6 and CMU_CLK7 are connected to the TIM filter counters.

### 25.8.1.1 CMU Block Diagram

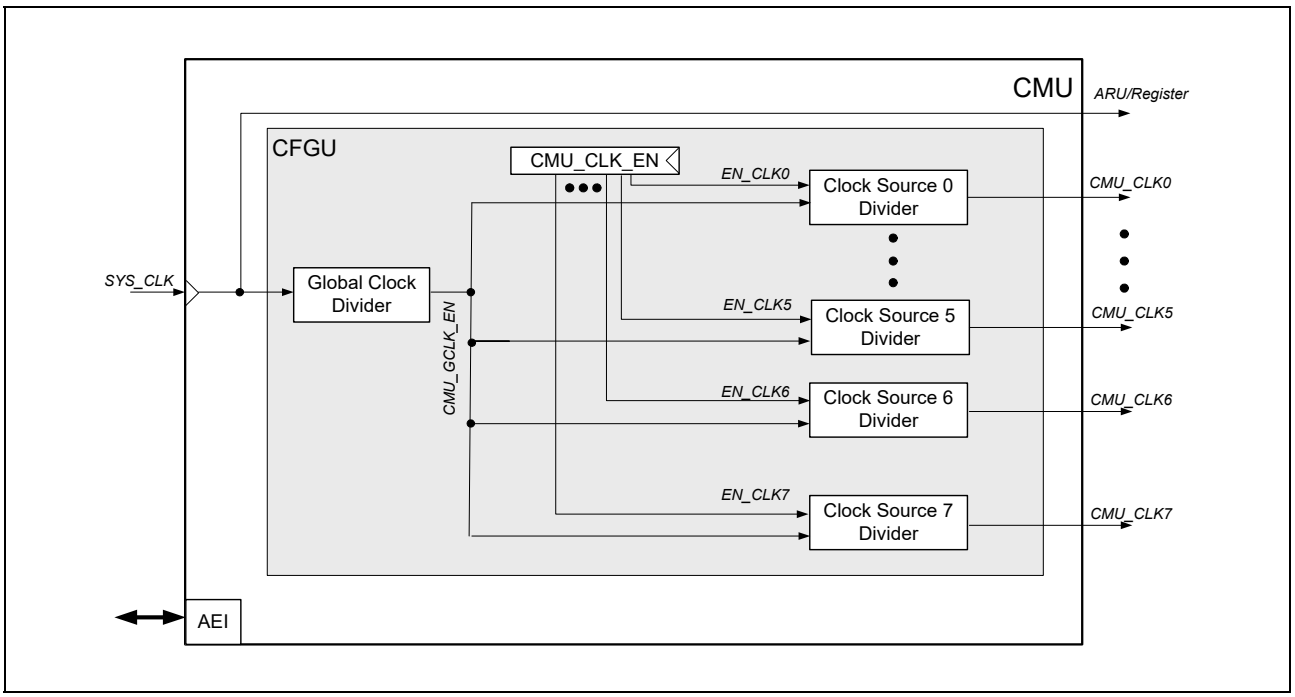



**Figure 25.17 CMU Block Diagram**

### 25.8.2 Global Clock Divider

The sub block Global Clock Divider can be used to divide the GTM-IP global input clock signal SYS_CLK into a common subdivided clock signal.

The divided clock signal of the sub block Global Clock Divider is implemented as an enable signal that enables dedicated clocks from the SYS_CLK signal to generate the user specified divided clock frequency.

The resulting fractional divider (Z/N) specified through equation:

$T_{CMU_GCLK_EN} = (Z/N)*T_{SYS_CLK}$

is implemented according the following algorithm

( Z: CMU_GCLK_NUM(23:0) ; N: GCLK_DEN(23:0) ; Z, N >0 ):

(1) Set remainder (R), operand1 (OP1) and operand2 (OP2) register during init-phase (with implicit conversion to signed):
R = Z, OP1 = N, OP2 = N-Z;

(2) After leaving init-phase (at least one CMU_CLK[x] has been enabled) the sign of remainder R for each SYS_CLK cycle will be checked:

(3) If R>0 keep updating remainder and keep CMU_GCLK_EN = ‘0’:
R = R-OP1;

(4) If R<0 update remainder and set CMU_GCLK_EN = ‘1’:
R = R-OP2;

After at most (Z/N+1) subtractions (3) there will be a negative R and an active phase of the generated clock enable (for one cycle) will be triggered (4). The remainder R is a measure for the distance to a real Z/N clock and will be regarded for the next generated clock enable cycle phase. The new R value will be R = R + (Z-N). In the worst case the remainder R will sum up to an additional cycle in the generated clock enable period after Z-cycles. In the other cases equally distributed additional cycles will be inserted for the generated clock enable. If Z is an integer multiple of N no additional cycles will be included for the generated clock enable at all.

Note that for a better resource sharing all arithmetic has been reduced to subtractions and the initialization of the remainder R uses the complement of (Z-N).

### 25.8.3 Configurable Clock Generation Subunit (CFGU)

The CMU subunit CFGU provides up to eight configurable clock divider blocks that divide the common CMU_GCLK_EN signal into dedicated enable signals for the GTM-IP sub blocks.

The configuration of the eight different clock signals CMU_CLK[x] (x: 0 to 7) always depends on the configuration of the global clock enable signal CMU_GCLK_EN. Additionally, each clock source has its own configuration data, provided by the control register GTM0CMUCLKxCTRL (x: 0 to 7).

According to the configuration of the Global Clock Divider, the configuration of the Clock Source x Divider is done by setting an appropriate value in the bit field CLK_CNT[x] of the register GTM0CMUCLKxCTRL.

The frequency fx =1/Tx of the corresponding clock enable signal CMU_CLK[x] can be determined by the unsigned representation of CLK_CNT[x] of the register GTM0CMUCLKxCTRL in the following way:

$T_{CMU_CLK[x]} = (CLK_CNT[x] + 1) * T_{CMU_GCLK_EN}$

The corresponding wave form is shown in **Figure 25.18**.

To avoid unexpected behavior of the hardware, the configuration of a register GTM0CMUCLKxCTRL can only be changed, when the corresponding clock signal CMU_CLK[x] is disabled.

Further, any changes to the registers GTM0CMUGCLKNUM and GTM0CMUGCLKDEN can only be performed, when all clock enable signals CMU_CLK[x] are disabled.

The clock source signals CMU_CLK[x] (x: 0 to 7) are implemented in form of enable signals for the corresponding registers, which means that the actual clock signal of all registers always use the SYS_CLK signal.

The hardware guarantees that all clock signals CMU_CLK[x], which were enabled simultaneous, are synchronized to each other. Simultaneous enabling does mean that the bits EN_CLK[x] in the register GTM0CMUCLKEN are set by the same write access.

### 25.8.4 Wave Form of Generated Clock Signal CMU_CLK[x]

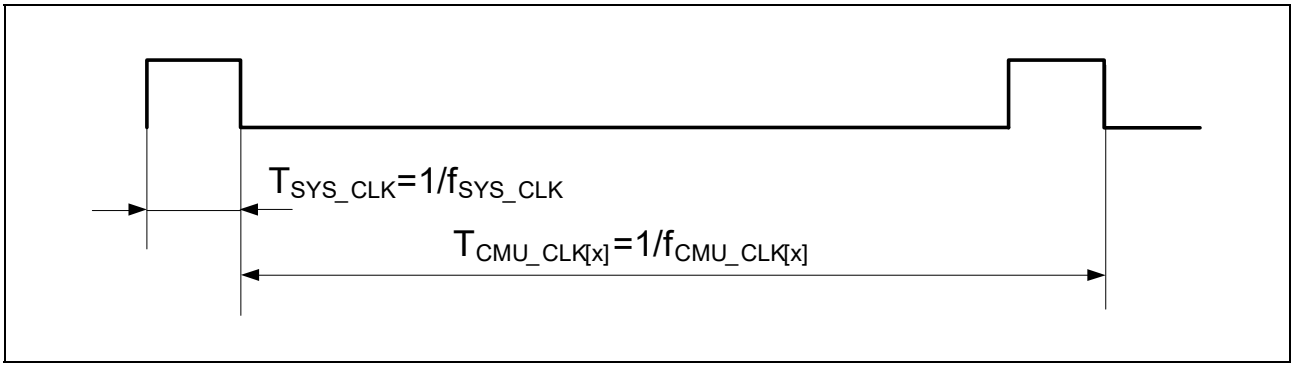



**Figure 25.18 Wave Form of Generated Clock Signal CMU_CLK[x]**

### 25.8.5 CMU Configuration Registers Overview

Following configuration registers are considered in CMU submodule:

**Table 25.48 Register List**

| Symbol | Register Name | Details in Section |
|---|---|---|
| GTM0CMUCLKEN | Clock enable | **25.8.6.1** |
| GTM0CMUGCLKNUM | Global clock control numerator | **25.8.6.2** |
| GTM0CMUGCLKDEN | Global clock control denominator | **25.8.6.3** |
| GTM0CMUCLK0CTRL | Control for clock source 0 | **25.8.6.4** |
| GTM0CMUCLK1CTRL | Control for clock source 1 | **25.8.6.4** |
| GTM0CMUCLK2CTRL | Control for clock source 2 | **25.8.6.4** |
| GTM0CMUCLK3CTRL | Control for clock source 3 | **25.8.6.4** |
| GTM0CMUCLK4CTRL | Control for clock source 4 | **25.8.6.4** |
| GTM0CMUCLK5CTRL | Control for clock source 5 | **25.8.6.4** |
| GTM0CMUCLK6CTRL | Control for clock source 6 | **25.8.6.4** |
| GTM0CMUCLK7CTRL | Control for clock source 7 | **25.8.6.4** |
| GTM0CMUGLBCTRL | Synchronizing ARU and clock source | **25.8.6.5** |

## 25.8.6 CMU Configuration Register Description

### 25.8.6.1 GTM0CMUCLKEN

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00300_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | EN_FXCLK | | EN_ECLK2 | | EN_ECLK1 | | EN_ECLK0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN_CLK7 | | EN_CLK6 | | EN_CLK5 | | EN_CLK4 | | EN_CLK3 | | EN_CLK2 | | EN_CLK1 | | EN_CLK0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.49 GTM0CMUCLKEN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23, 22 | EN_FXCLK | Enable all CMU_FXCLK, see bits 1, 0<br>**NOTE**<br>EN_ECLK0-2 and EN_FXCLK not implemented for this device. Writing values unequal 0 will result in AEI_STATUS signal "10" |
| 21, 20 | EN_ECLK2 | Enable ECLK 2 generation subunit, see bits 1, 0 |
| 19, 18 | EN_ECLK1 | Enable ECLK 1 generation subunit, see bits 1, 0 |
| 17, 16 | EN_ECLK0 | Enable ECLK 0 generation subunit, see bits 1, 0 |
| 15, 14 | EN_CLK7 | Enable clock source 7, see bits 1, 0 |
| 13, 12 | EN_CLK6 | Enable clock source 6, see bits 1, 0 |
| 11, 10 | EN_CLK5 | Enable clock source 5, see bits 1, 0 |
| 9, 8 | EN_CLK4 | Enable clock source 4, see bits 1, 0 |
| 7, 6 | EN_CLK3 | Enable clock source 3, see bits 1, 0 |
| 5, 4 | EN_CLK2 | Enable clock source 2, see bits 1, 0 |
| 3, 2 | EN_CLK1 | Enable clock source 1, see bits 1, 0 |
| 1, 0 | EN_CLK0 | Enable clock source 0<br>00: Clock source is disabled (ignore write access)<br>01: Disable clock signal and reset internal states<br>10: Enable clock signal<br>11: Clock signal enabled (ignore write access)<br>**NOTES**<br>1. Any read access to an EN_CLK[x] bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored.<br>2. Any disabling to EN_CLK[x] will be reset internal counters for configurable clocks. |

### 25.8.6.2 GTM0CMUGCLKNUM

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00304_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CMU_GCLK_NUM | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMU_GCLK_NUM | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.50 GTM0CMUGCLKNUM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | CMU_GCLK_NUM | Numerator for global clock divider. Defines numerator of the fractional divider.<br>**NOTES**<br>1. Value can only be modified when all clock enables EN_CLK[x] are disabled.<br>2. The CMU hardware alters the content of GTM0CMUGCLKNUM and GTM0CMUGCLKDEN automatically to $1_H$, if GTM0CMUGCLKNUM is specified less than GTM0CMUGCLKDEN or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register GTM0CMUGCLKNUM followed by a single write to register GTM0CMUGCLKDEN. |

### 25.8.6.3 GTM0CMUGCLKDEN

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00308_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | GCLK_DEN | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GCLK_DEN | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.51 GTM0CMUGCLKDEN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | GCLK_DEN | Denominator for global clock divider. Defines denominator of the fractional divider<br>**NOTES**<br>1. Value can only be modified when all clock enables EN_CLK[x] are disabled.<br>2. The CMU hardware alters the content of GTM0CMUGCLKNUM and GTM0CMUGCLKDEN automatically to 0x1, if GTM0CMUGCLKNUM is specified less than GTM0CMUGCLKDEN or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register GTM0CMUGCLKNUM followed by a single write to register GTM0CMUGCLKDEN. |

### 25.8.6.4 GTM0CMUCLKxCTRL (x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0CMUCLK0CTRL: <GTM_base> + $0030C_H$
GTM0CMUCLK1CTRL: <GTM_base> + $00310_H$
GTM0CMUCLK2CTRL: <GTM_base> + $00314_H$
GTM0CMUCLK3CTRL: <GTM_base> + $00318_H$
GTM0CMUCLK4CTRL: <GTM_base> + $0031C_H$
GTM0CMUCLK5CTRL: <GTM_base> + $00320_H$
GTM0CMUCLK6CTRL: <GTM_base> + $00324_H$
GTM0CMUCLK7CTRL: <GTM_base> + $00328_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CLK_CNT | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLK_CNT | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.52 GTM0CMUCLKxCTRL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | CLK_CNT | Clock count. Defines count value for the clock divider of clock source CMU_CLK[x] (x = 0 to 7).<br>**NOTE**<br>Value can only be modified when clock enable EN_CLK[x] (x = 0 to 7) is disabled. |

### 25.8.6.5 GTM0CMUGLBCTRL

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00348_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ARU_ADDR_RSTGLB |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 25.53 GTM0CMUGLBCTRL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 0 | ARU_ADDR_RSTGLB | Reset ARU address counter<br>**NOTES**<br>1. Writing value “1” to this bit field results in a request to reset of the ARU address counter. The next following write access to register GTM0CMUCLKEN applies the ARU reset and resets this bit. This feature can be used to synchronize the ARU round trip time to the CMU clocks.<br>2. This bit is write protected. Before writing to this bit set bit RF_PROT of register GTM0GTMCTRL to '0'. |

# 25.9 Time Base Unit (TBU)

## 25.9.1 Overview

The Time Base Unit TBU provides common time bases for the GTM-IP. The TBU submodule is organized in channels, where the number of channels is device dependent. There are at most two channels implemented inside the TBU. The TBU channel 0 time base register GTM0TBU0BASE is 27 bits and it is configurable whether the lower 24 bit or the upper 24 bit are provided to the GTM as signal TBU_TS0. The TBU channels 1 has a time base register GTM0TBU1BASE of 24 bit length. The time base register value TBU_TS1 are provided to subsequent submodules of the GTM.

The TBU_UP1 signals are set to high for a single SYS_CLK period, whenever the corresponding signal TBU_TS1 is getting updated. The signal TBU_UP0_L is set to high for a single SYS_CLK period if the signal TBU_TS0 and TBU_TS1 is getting updated and TBU_UP0_H is set to high for a single SYS_CLK period, whenever if the upper 24 bit of TBU_TS0 are updated.

The time base channels can run independently of each other and can be enabled and disabled synchronously by control bits in a global TBU channel enable register GTM0TBUCHEN. **Section 25.9.1.1, TBU Block Diagram** shows a block diagram of the Time Base Unit.

### 25.9.1.1 TBU Block Diagram

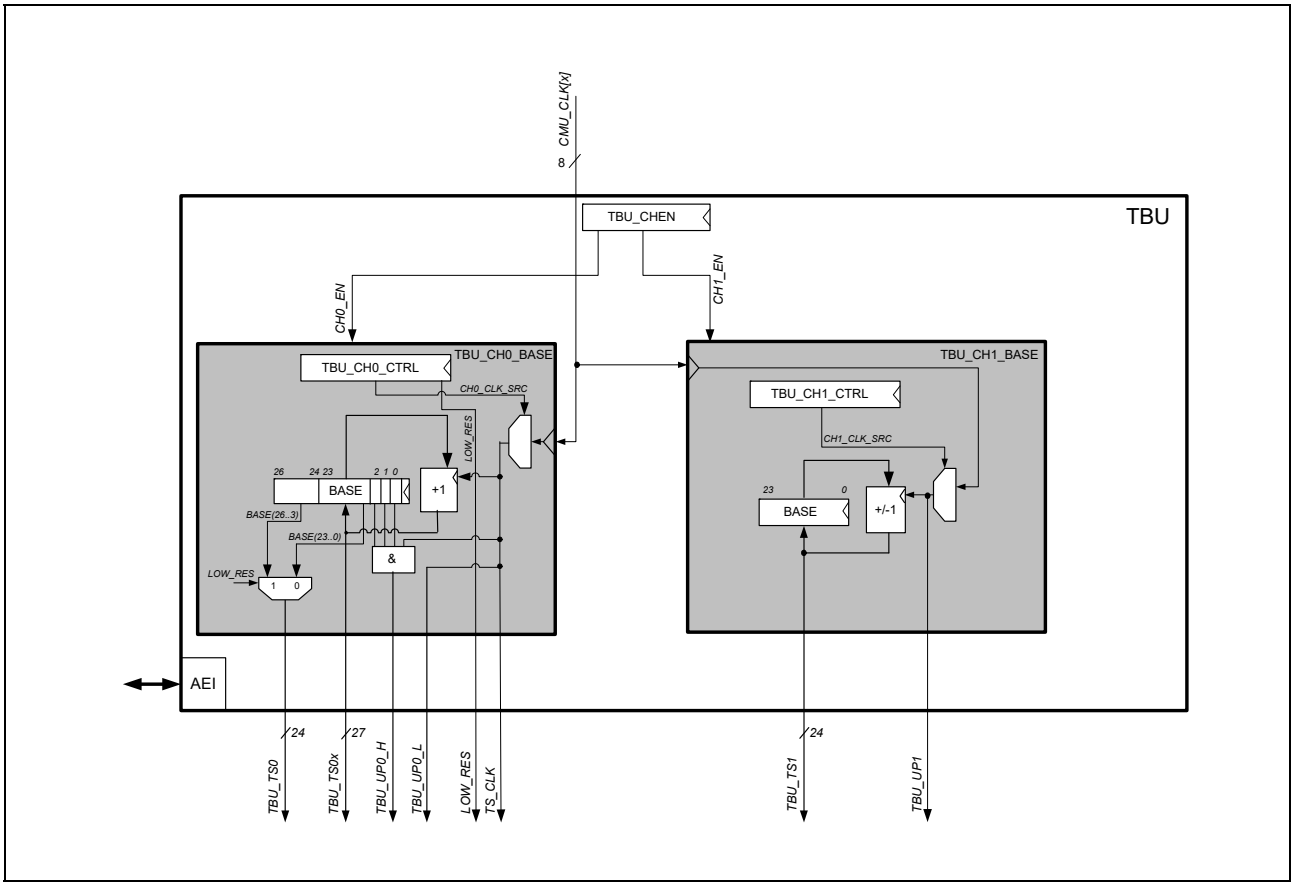



Figure 25.19 BU Block Diagram

The configuration of the independent time base channels TBU_BASE_[z] (z = 0, 1) is done via the AEI interface. Each TBU channel may select one of the eight CMU_CLK[x] (x = 0 to 7) signals coming from the CMU submodule.

## 25.9.2 TBU Time Base Channels

The time base values are generated within the TBU time base channels in two independent operation modes.

### 25.9.2.1 TBU Channel Modes

TBU channel 0 provides a 27 bit counter in a free running counter mode. Dependent on the bit field LOW_RES of register GTM0TBU0CTRL, the lower 24 bits (bit 0 to 23) or the upper 24 bits (bits 3 to 26) are provided to the GTM submodules.

TBU channel 1 is running as a free running counter.

The time base register GTM0TBU1BASE can be initialized with a start value just before enabling the corresponding TBU channel.

Moreover, the time base register GTM0TBU1BASE can always be read in order to determine the actual value of the counter.

The time base register GTM0TBU1BASE is updated on every specified incoming clock event by the selected signal CMU_CLK[x] (dependent on GTM0TBUzCTRL register). In general the time base register GTM0TBU1BASE is incremented on every CMU_CLK[x] clock tick.

## 25.9.3 TBU Configuration Registers Overview

Following table shows a conclusion of configuration registers address offsets and initial values.

**Table 25.54 Register List**

| Symbol | Register Name | Details in Section |
|---|---|---|
| GTM0TBUCHEN | TBU global channel enable | **25.9.4.1** |
| GTM0TBU0CTRL | TBU channel 0 control | **25.9.4.2** |
| GTM0TBU0BASE | TBU channel 0 base | **25.9.4.3** |
| GTM0TBU1CTRL | TBU channel 1 control | **25.9.4.4** |
| GTM0TBU1BASE | TBU channel 1 base | **25.9.4.5** |

## 25.9.4 TBU Registers description

### 25.9.4.1 GTM0TBUCHEN

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00100_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ENDIS_CH2 | | ENDIS_CH1 | | ENDIS_CH0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.55 GTM0TBUCHEN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5, 4 | ENDIS_CH2 | TBU channel 2 enable/disable control. See bits 1, 0<br>**NOTE**<br>ENDIS_CH2 not implemented for this device. Writing values unequal 0 will result in AEI_STATUS signal “10” |
| 3, 2 | ENDIS_CH1 | TBU channel 1 enable/disable control. See bits 1, 0 |
| 1, 0 | ENDIS_CH0 | TBU channel 0 enable/disable control.<br>Write of following double bit values is possible:<br>00: Don’t care, bits 1:0 will not be changed<br>01: Channel disabled: is read as 00 (see below)<br>10: Channel enabled: is read as 11 (see below)<br>11: Don’t care, bits 1:0 will not be changed<br>**NOTE**<br>Read of following double values means:<br>00: Channel disabled<br>11: Channel enabled |

### 25.9.4.2 GTM0TBU0CTRL

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00104_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CH_CLK_SRC | | | LOW_RES |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 25.56 GTM0TBU0CTRL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 3 to 1 | CH_CLK_SRC | Clock source for channel x (x = 0 to 2) time base counter<br>000: CMU_CLK0 selected<br>001: CMU_CLK1 selected<br>010: CMU_CLK2 selected<br>011: CMU_CLK3 selected<br>100: CMU_CLK4 selected<br>101: CMU_CLK5 selected<br>110: CMU_CLK6 selected<br>111: CMU_CLK7 selected<br>**NOTE**<br>This value can only be modified if channel 0 is disabled. |
| 0 | LOW_RES | GTM0TBU0BASE register resolution.<br>0: TBU channel uses lower counter bits (bit 0 to 23)<br>1: TBU channel uses upper counter bits (bit 3 to 26)<br>**NOTES**<br>1. The two resolutions for the TBU channel 0 can be used in the TIM channel 0.<br>2. This value can only be modified if channel 0 is disabled. |

### 25.9.4.3 GTM0TBU0BASE

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00108_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | BASE | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BASE | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.57 GTM0TBU0BASE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 26 to 0 | BASE | Time base value for channel 0.<br>**NOTES**<br>1. The value of BASE can only be written if the TBU channel 0 is disabled.<br>2. If channel 0 is enabled, a read access to this register provides the current value of the underlying 27 bit counter. |

### 25.9.4.4 GTM0TBU1CTRL

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $0010C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CH_CLK_SRC | | | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R |

**Table 25.58 GTM0TBU1CTRL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 3 to 1 | CH_CLK_SRC | Clock source for channel 1 time base counter<br>000: CMU_CLK0 selected<br>001: CMU_CLK1 selected<br>010: CMU_CLK2 selected<br>011: CMU_CLK3 selected<br>100: CMU_CLK4 selected<br>101: CMU_CLK5 selected<br>110: CMU_CLK6 selected<br>111: CMU_CLK7 selected<br>**NOTE**<br>This value can only be modified if channel y was disabled. |
| 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

### 25.9.4.5 GTM0TBU1BASE

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00110_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | BASE | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BASE | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.59 GTM0TBU1BASE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | BASE | Time base value for channel 1<br>**NOTES**<br>1. The value of BASE can only be written if the corresponding TBU channel y is disabled.<br>2. If the corresponding channel y is enabled, a read access to this register provides the current value of the underlying counter. |

# 25.10 Timer Input Module (TIM)

## 25.10.1 Overview

The Timer Input Module (TIM) is responsible for filtering and capturing input signals of the GTM. Several characteristics of the input signals can be measured inside the TIM channels. For advanced data processing the detected input characteristics of the TIM module can be routed through the ARU to subsequent processing units of the GTM.

Input characteristics mean either time stamp values of detected input rising or falling edges together with the new signal level or the number of edges received since channel enable together with the actual time stamp or PWM signal durations for a whole PWM period.

The architecture of TIM is shown in **Figure 25.20**.

### 25.10.1.1 TIM Block Diagram

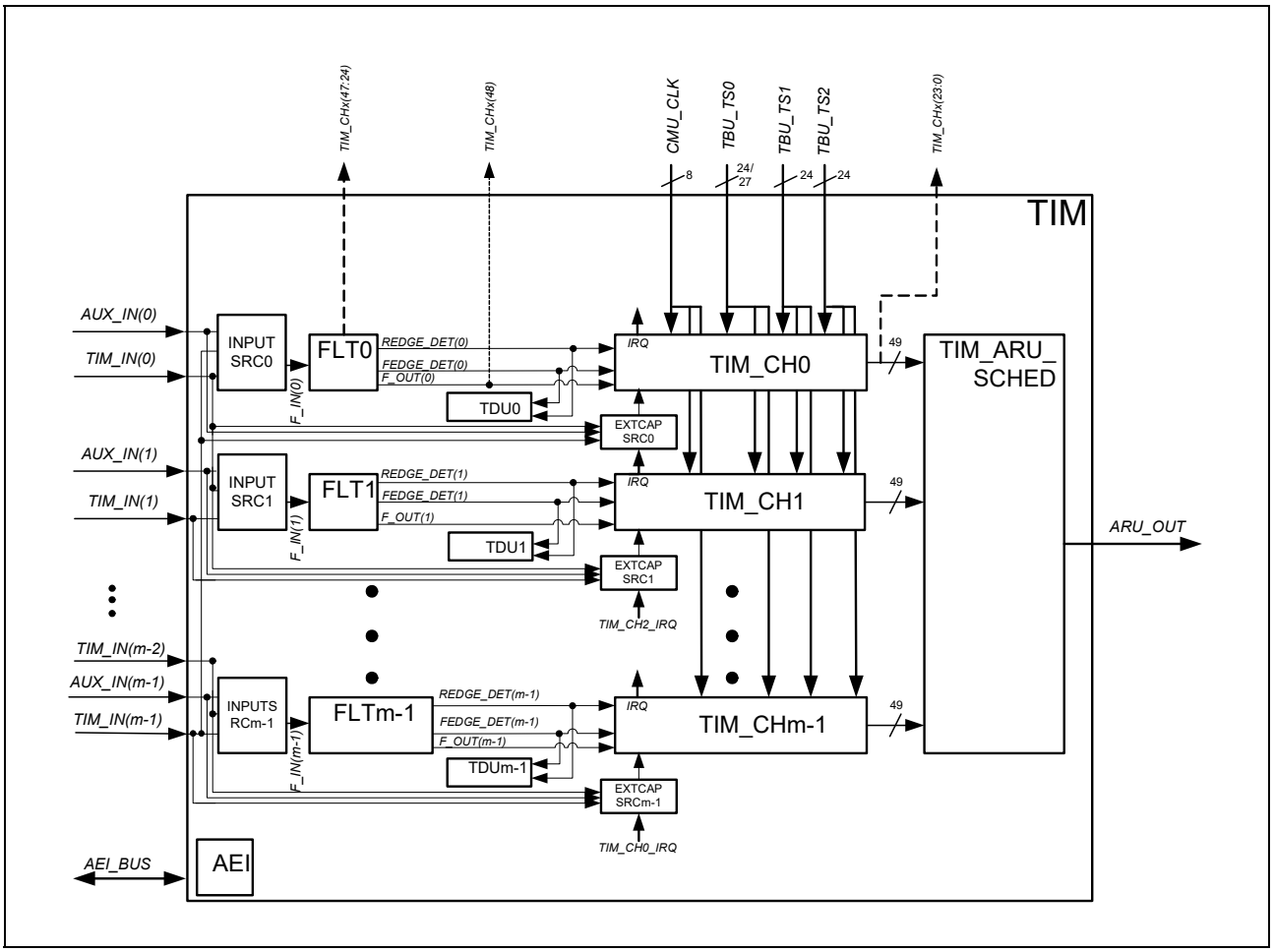



**Figure 25.20 TIM Block Diagram**

The number of channels m inside a TIM submodule depends on the device.

Each of the *m* dedicated input signals are filtered inside the FLTx subunit of the TIM Module. It should be noted that the incoming input signals are synchronized to the clock SYS_CLK, resulting in a delay of two SYS_CLK periods for the incoming signals.

The submodule TIM provides different filter mechanisms described in more detail in **Section 25.10.2, TIM Filter Functionality (FLT)** . After filtering, the signal is routed to the corresponding TIM channel.

The measurement values can be read by the CPU directly via the AEI-Bus or they can be routed through the ARU to other submodules of the GTM.

For timeout detection of an incoming signal (no subsequent edge detected during a specified duration) each individual channel has a Timeout Detection Unit (TDU).

The two (three) time bases coming from the TBU are connected to the TIM channels to annotate time stamps to incoming signals. For TIM0 the extended 27 bit width time base TBU_TS0 is connected to the TIM channels, and the user has to select if the lower 24 bits (TBU_TS0[23:0]) or the higher 24 bits (TBU_TS0[26:3]) are stored inside the GPR0 and GPR1 registers.

### 25.10.1.2 Input source selection INPUTSRCx

It can be configured which source shall be used for processing in the FLT,TDU,TIM_CH units. It can be selected by the bit fields CICTRL and MODE_x, VAL_x in the register GTM0TIM0INSRC which source is in use.

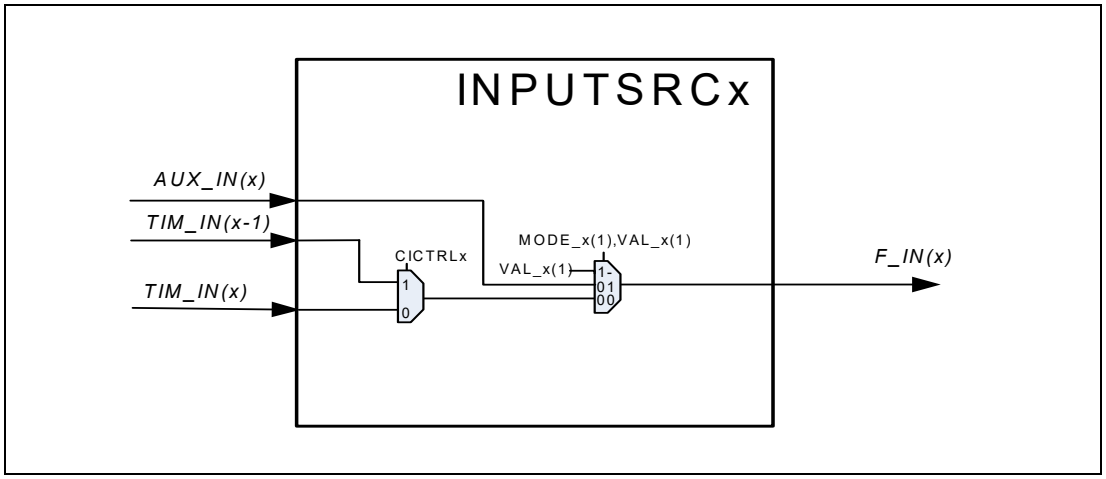



**Figure 25.21 INPUTSRC Block Diagram**

In a certain MODE_x, VAL_x combination the input signal F_IN(x) can be driven by VAL_x(1) with 0 or 1 directly.

Due to the fact that all 8 channels are bundled in the register GTM0TIM0INSRC a synchronous control of all 8 input channels is possible.

Two adjacent channels can be combined by setting the CICTRL bit field in the corresponding GTM0TIMixCTRLregister. This allows for a combination of complex measurements on one input signal with two TIM channels.

The additional input signal AUX_IN[x] can be selected as an input signal. See **Section 25.6.1.1, GTM Architecture Block Diagram** .

### 25.10.1.3 Input observation

It is possible to observe for all channels of one instance by reading TIM_INP_VAL the actual signal values of the following processing stages:

- TIM_IN(7:0) signals after TIM input synchronization
- TIM F_IN(7:0) signals after TIM INPUTSRC selection (input to TIM_FLT)
- TIM F_OUT(7:0) signals after TIM filter functionality (output of TIM_FLT)

### 25.10.1.4 External capture source selection EXTCAPSRCx

Each channel can operate on an external capture signal EXT_CAPTURE. The source to use for this signal can be configured by the bit field EXT_CAP_SRCx in the register GTM0TIM00ECTRL .

The EXT_CAPTURE signal will be distributed to other GTM modules (e.g. TOM/ATOM) .

**NOTE**

The EXT_CAPTURE signal generation is independent of the signal EXT_CAP_EN.

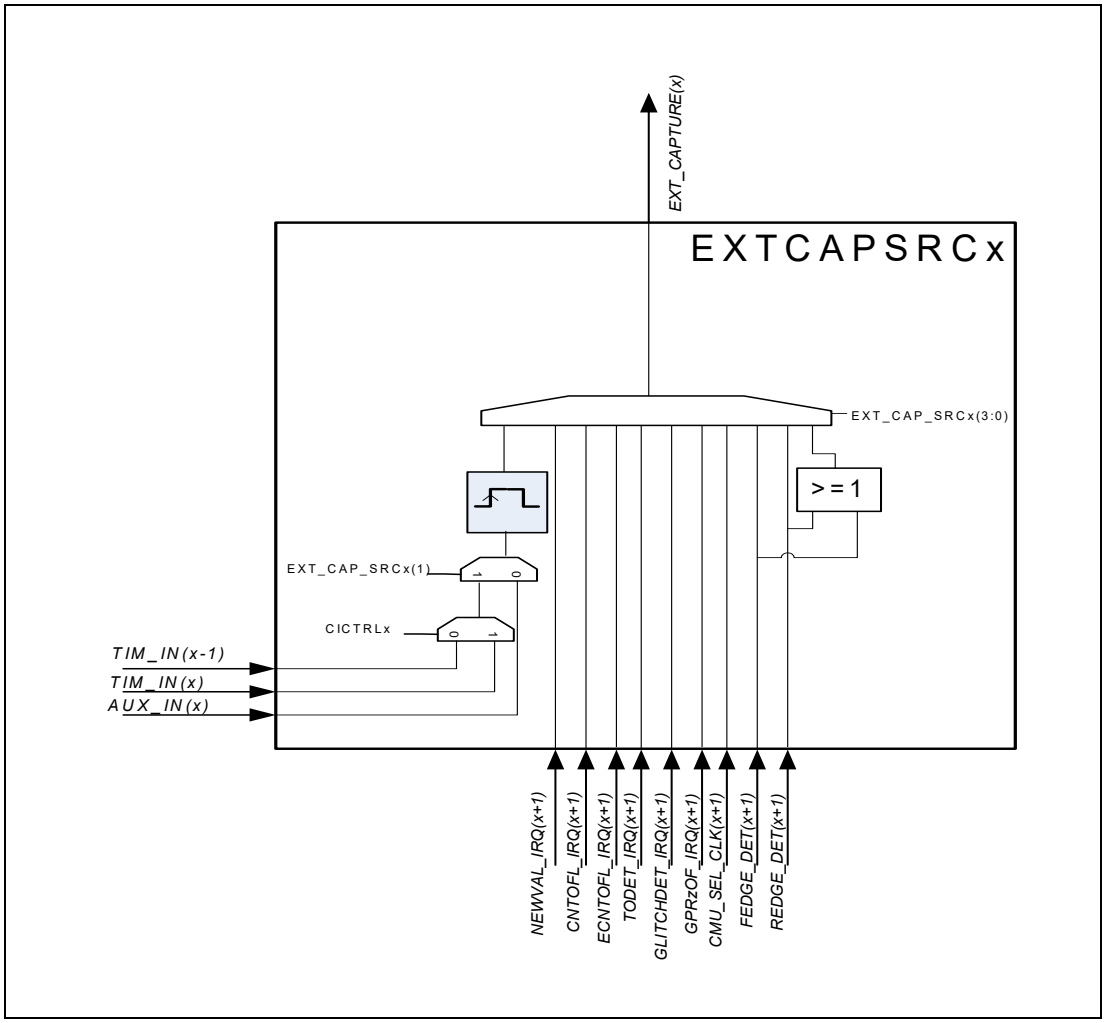



**Figure 25.22 XTCAPSRC Block Diagram**

The external capture functionality can be enabled for the TIM channel x with the bit EXT_CAP_EN in the register GTM0TIMixCTRL, it will trigger on each rising edge. A pulse generation for each rising edge of the selected input signal TIM_IN[x] and AUX_IN[x] is applied.

The six TIM channel interrupt sources can be triggered by the operation in the certain TIM channel modes. Alternatively they can be issued by a soft trigger using the corresponding bits in the register GTM0TIMixIRQFORCINT.

## 25.10.2 TIM Filter Functionality (FLT)

### 25.10.2.1 Overview

The TIM submodule provides a configurable filter mechanism for each input signal. These filter mechanism is provided inside the FLT subunit.

FLT architecture is shown in **Figure 25.23**.

The filter includes a clock synchronization unit (CSU), an edge detection unit (EDU), and a filter counter associated to the filter unit (FLTU).

The CSU is synchronizing the incoming signal F_IN to the selected filter clock frequency, which is controlled with the bit field FLT_CNT_FRQ of register GTM0TIMixCTRL.

The synchronized input signal F_IN_SYNC is used for further processing within the filter.

It should be noted that glitches with a duration less than the selected CMU clock period are lost.

The filter modes can be applied individually to the falling and rising edges of an input signal. The following filter modes are available:

- Immediate edge propagation mode,
- Individual de-glitch time mode (up/down counter)
- Individual de-glitch time mode (hold counter).

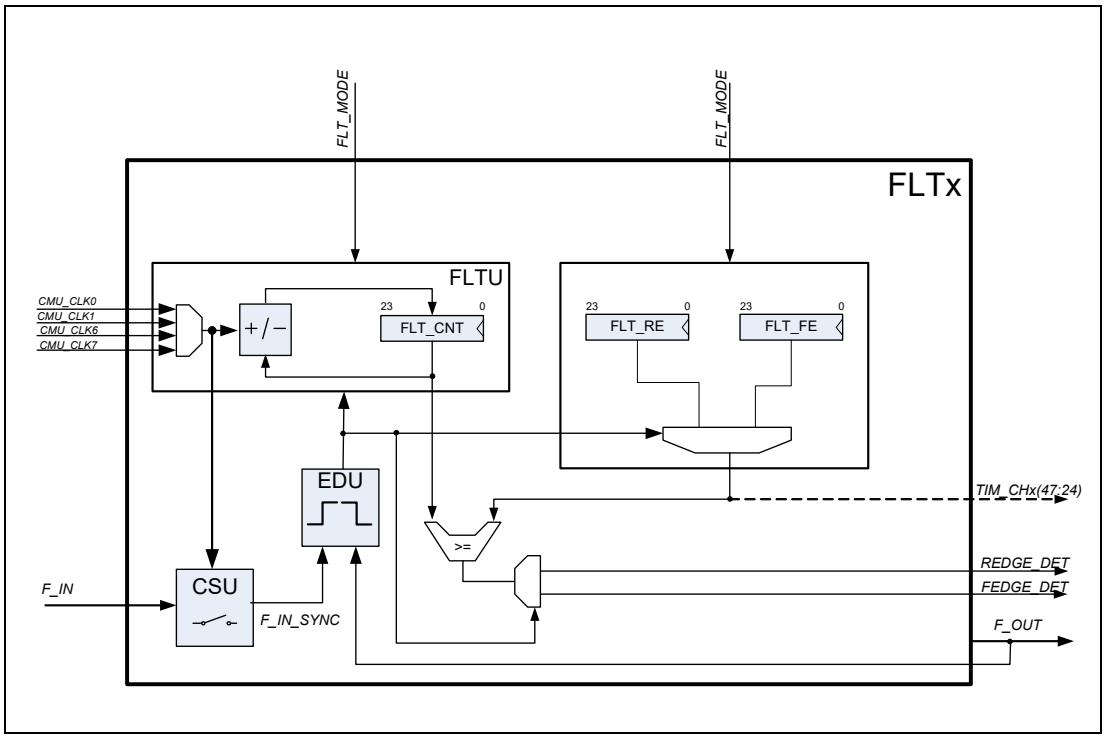



**Figure 25.23 FLT Architecture**

The filter parameters (deglitch and acceptance time) for the rising and falling edge can be configured inside the two filter parameter registers FLT_RE (rising edge) and FLT_FE (falling edge). The exact meaning of the parameter depends on the filter mode.

However the delay time T of both filter parameters FLT_xE can always be determined by:

$T = (FLT_xE + 1) * T_{FLT_CLK}$,

whereas $T_{FLT_CLK}$ is the clock period of the selected CMU clock signal in bit field FLT_CNT_FRQ of register GTM0TIMixCTRL.

When a glitch is detected on an input signal a status flag GLITCHDET is set inside the GTM0TIMixIRQNOTIFY register.

**Table 25.60** gives an overview about the meanings for the registers FLT_RE and FLT_FE. In the individual deglitch time modes, the actual filter threshold for a detected regular edge is provided on the TIM[i]_CH[x] (47:24) output line. In the case of immediate edge propagation mode, a value of zero is provided on the TIM[i]_CH[x] (47:24) output line.

**Table 25.60 Filter Parameter summary for the different Filter Modes**

| Filter mode | Meaning of FLT_RE | Meaning of FLT_FE |
|---|---|---|
| Immediate edge propagation | Acceptance time for rising edge | Acceptance time for falling edge |
| Individual de-glitch time (up/ down counter) | De-glitch time for rising edge | De-glitch time for falling edge |
| Individual de-glitch time (hold counter) | De-glitch time for rising edge | De-glitch time for falling edge |

A counter FLT_CNT is used to measure the glitch and acceptance times.

The frequency of the FLT_CNT counter is configurable in bit field FLT_CNT_FRQ of register GTM0TIMixCTRL.

The counter FLT_CNT can either be clock with the CMU_CLK0, CMU_CLK1, CMU_CLK6 or the CMU_CLK7 signal. These signals are coming from the CMU submodule.

The FLT_CNT, FLT_FE and FLT_RE registers are 24-bit width. For example, when the resolution of the CMU_CLK0 signal is 50ns this allows maximal de-glitch and acceptance times of about 838ms for the filter.

### 25.10.2.2 TIM Filter Modes

#### (1) Immediate Edge Propagation Mode

In immediate edge propagation mode after detection of an edge the new signal level on F_IN_SYNC is propagated to F_OUT with a delay of one TFLT_CLK period and the new signal level remains unchanged until the configured acceptance time expires.

For each edge type the acceptance time can be specified separately in the FLT_RE and FLT_FE registers.

Each signal change on the input F_IN_SYNC during the duration of the acceptance time has no effect on the output signal level F_OUT of the filter but it sets the glitch GLITCHDET bit in the GTM0TIMixIRQNOTIFY register.

After it expires an acceptance time the input signal F_IN_SYNC is observed and on signal level change the filter raises a new detected edge and the new signal level is propagated to F_OUT.

Independent of a signal level change the value of F_OUT is always set to F_IN_SYNC, when the acceptance time expires (see also **Figure 25.25** ).

**Figure 25.24** shows an example for the immediate edge propagation mode, in the case of rising edge detection. Both, the signal before filtering (F_IN) and after filtering (F_OUT) are shown. The acceptance time at1 is specified in the register FLT_RE.

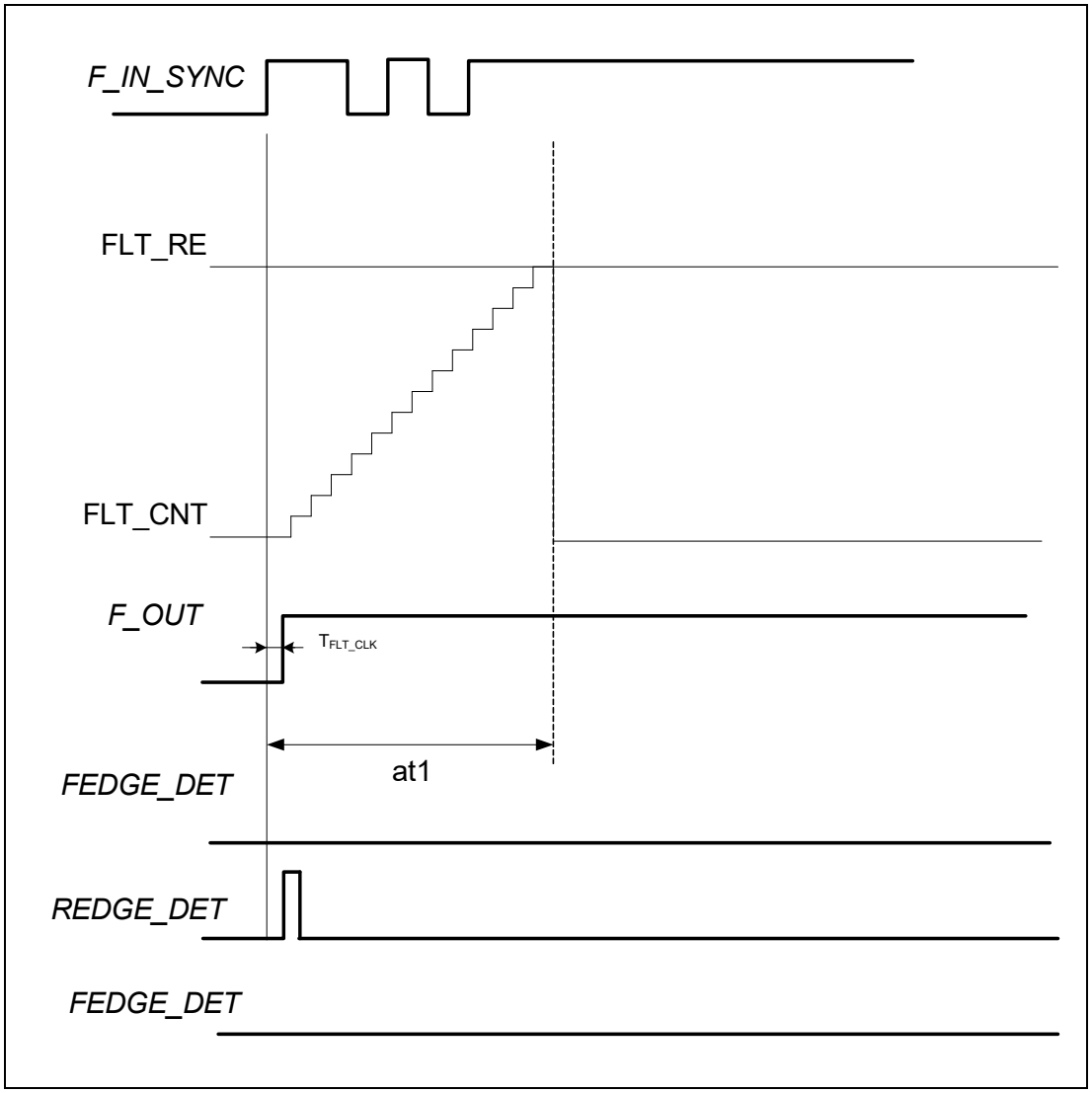



**Figure 25.24 Immediate Edge Propagation Mode in the case of a rising edge**

In immediate edge propagation mode the glitch measurement mechanism is not applied to the edge detection. Detected edges on F_IN_SYNC are transferred directly to F_OUT.

The counter FLT_CNT is incremented until acceptance time threshold is reached.

**Figure 25.25** shows a more complex example of the TIM filter, in which both, rising and falling edges are configured in immediate edge propagation mode.

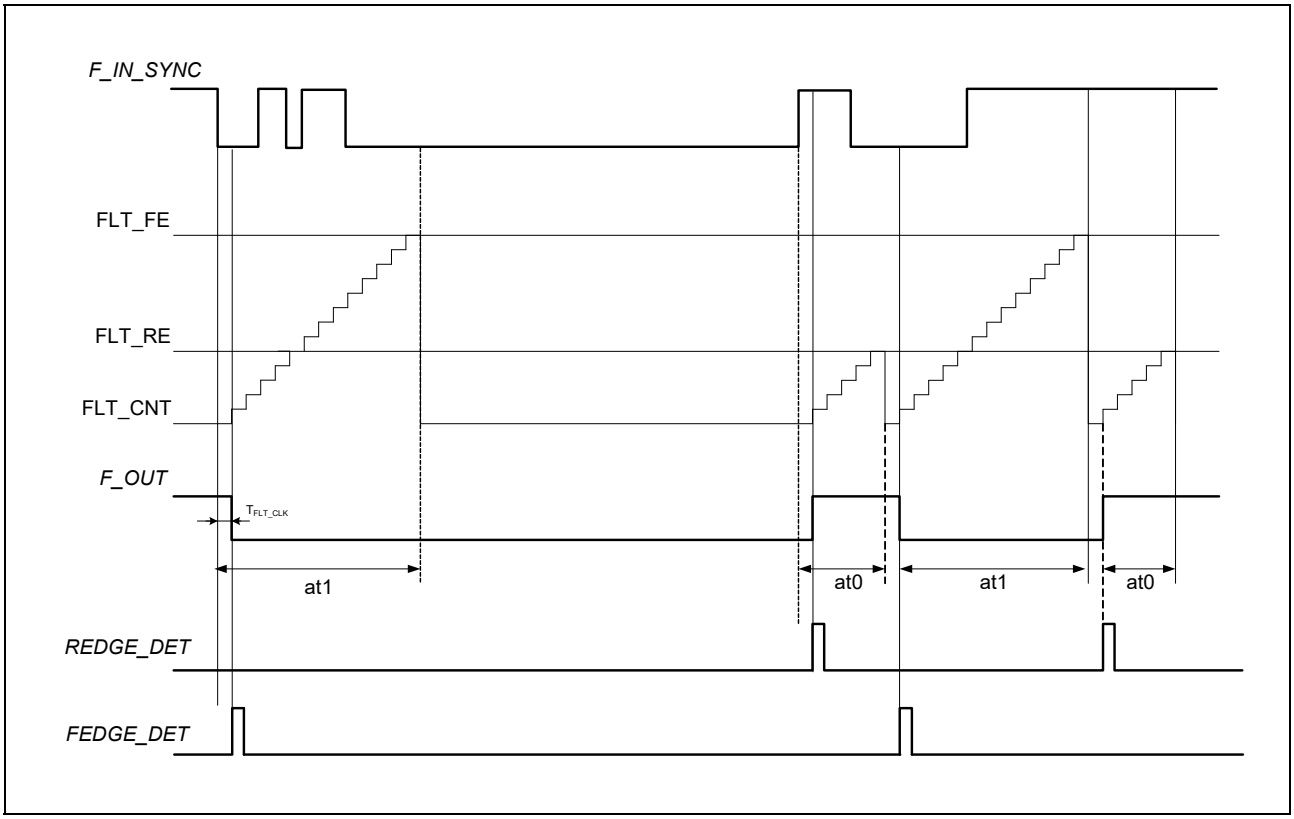



**Figure 25.25 Immediate Edge Propagation Mode in the case of a rising and falling edge**

If the FLT_CNT has reached the acceptance time for a specific signal edge and the signal F_IN_SYNC has already changed to the opposite level of F_OUT, the opposite signal level is set to F_OUT and the acceptance time measurement is started immediately. **Figure 25.25** shows this scenario at the detection of the first rising edge and the second falling edge.

### (2) Individual De-Glitch Time Mode (up/down counter)

In individual de-glitch time mode (up/down counter) each edge of an input signal can be filtered with an individual de-glitch threshold filter value mentioned in the registers FLT_RE and FLT_FE, respectively.

The filter counter register FLT_CNT is incremented when the signal level on F_IN_SYNC is unequal to the signal level on F_OUT and decremented if F_IN_SYNC equals F_OUT.

After FLT_CNT has reached a value of zero during decrementation the counter is stopped immediately.

If a glitch is detected a glitch detection bit GLITCHDET is set in the GTM0TIMixIRQNOTIFY register.

The detected edge signal together with the new signal level is propagated to F_OUT after the individual de-glitch threshold is reached. **Figure 25.26** shows the behavior of the filter in individual de-glitch time (up/down counter) mode in the case of the rising edge detection.

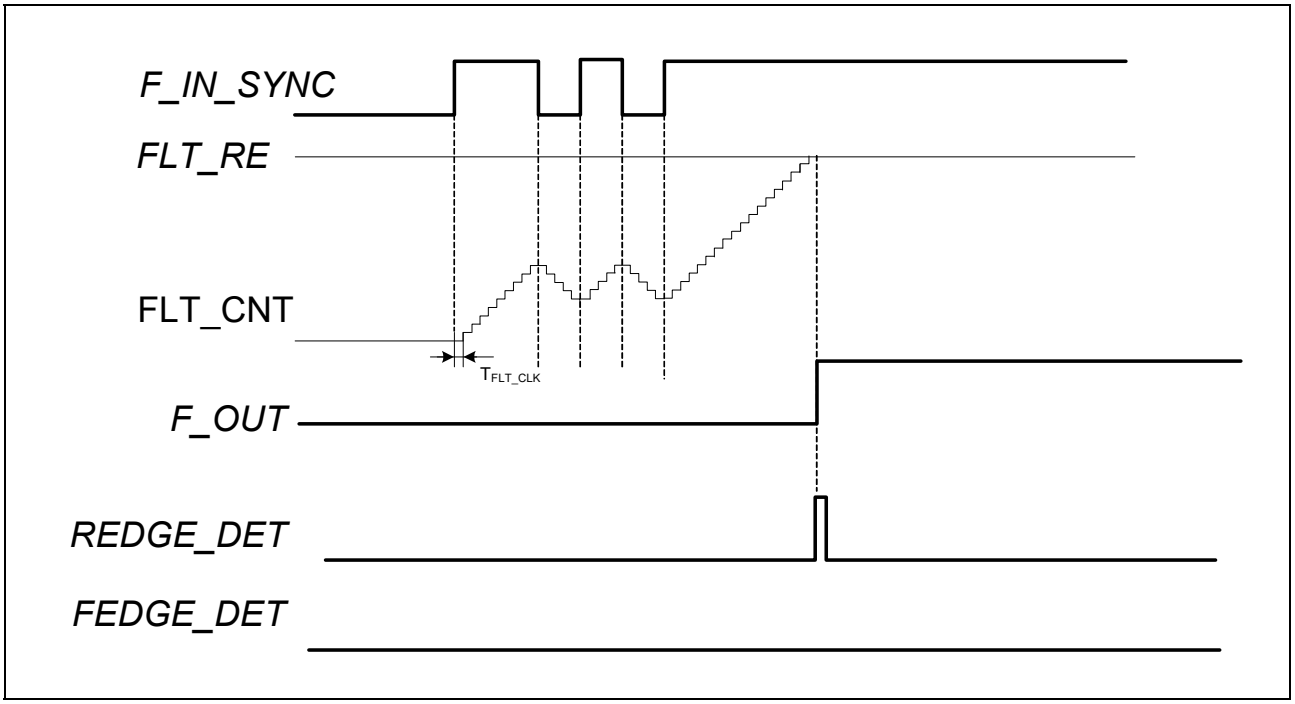



**Figure 25.26 Individual De-Glitch Time Mode (up/down counter) in the case of a rising edge**

### (3) Individual De-Glitch Time Mode (hold counter)

In individual de-glitch time mode (hold counter) each edge of an input signal can be filtered with an individual de-glitch threshold filter value mentioned in the registers FLT_RE and FLT_FE, respectively.

The filter counter register FLT_CNT is incremented when the signal level on F_IN_SYNC is unequal to the signal level on F_OUT and the counter value of FLT_CNT is hold if F_IN equals F_OUT.

If a glitch is detected the glitch detection bit GLITCHDET is set in the GTM0TIMixIRQNOTIFY register.

The detected edge signal together with the new signal level is propagated to F_OUT after the individual de-glitch threshold is reached. **Figure 25.27** shows the behavior of the filter in individual de-glitch time (hold counter) mode in the case of the rising edge detection.

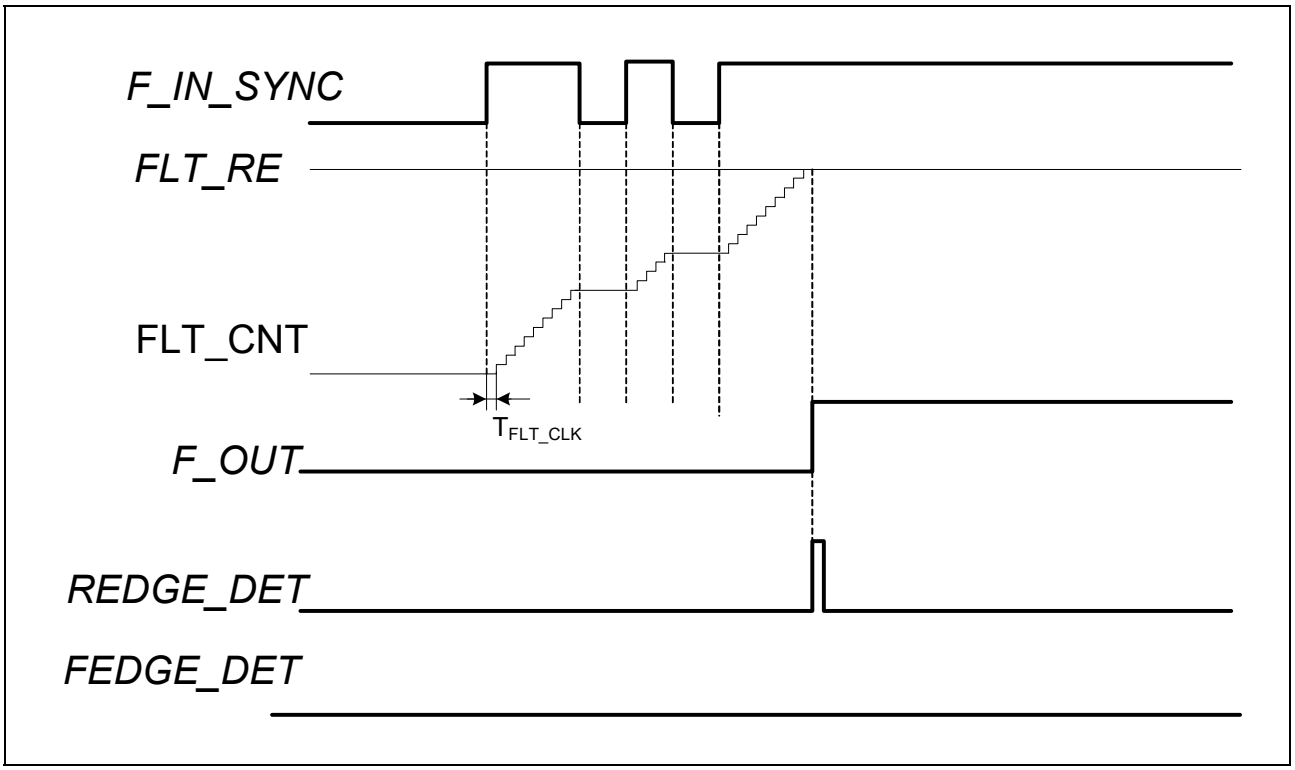



**Figure 25.27 Individual De-Glitch Time Mode (hold counter) in the case of a rising edge**

#### (4) Immediate Edge Propagation and Individual De-Glitch Mode

As already mentioned, the three different filter modes can be applied individually to each edge of the measured signal.

However, if one edge is configured with immediate edge propagation and the other edge with an individual deglitch mode (whether up/down counter or hold counter) a special consideration has to be applied.

Assume that the rising edge is configured for immediate edge propagation and the falling edge with individual deglitch mode (up/down counter) as shown in **Figure 25.28**.

If the falling edge of the incoming signal already occurs during the measuring of the acceptance time of the rising edge, the measurement of the deglitch time on the falling edge is started delayed, but immediately after the acceptance time measurement phase of the rising edge has finished.

Consequently, the deglitch counter can not measure the time TERROR, as shown in **Figure 25.28**.

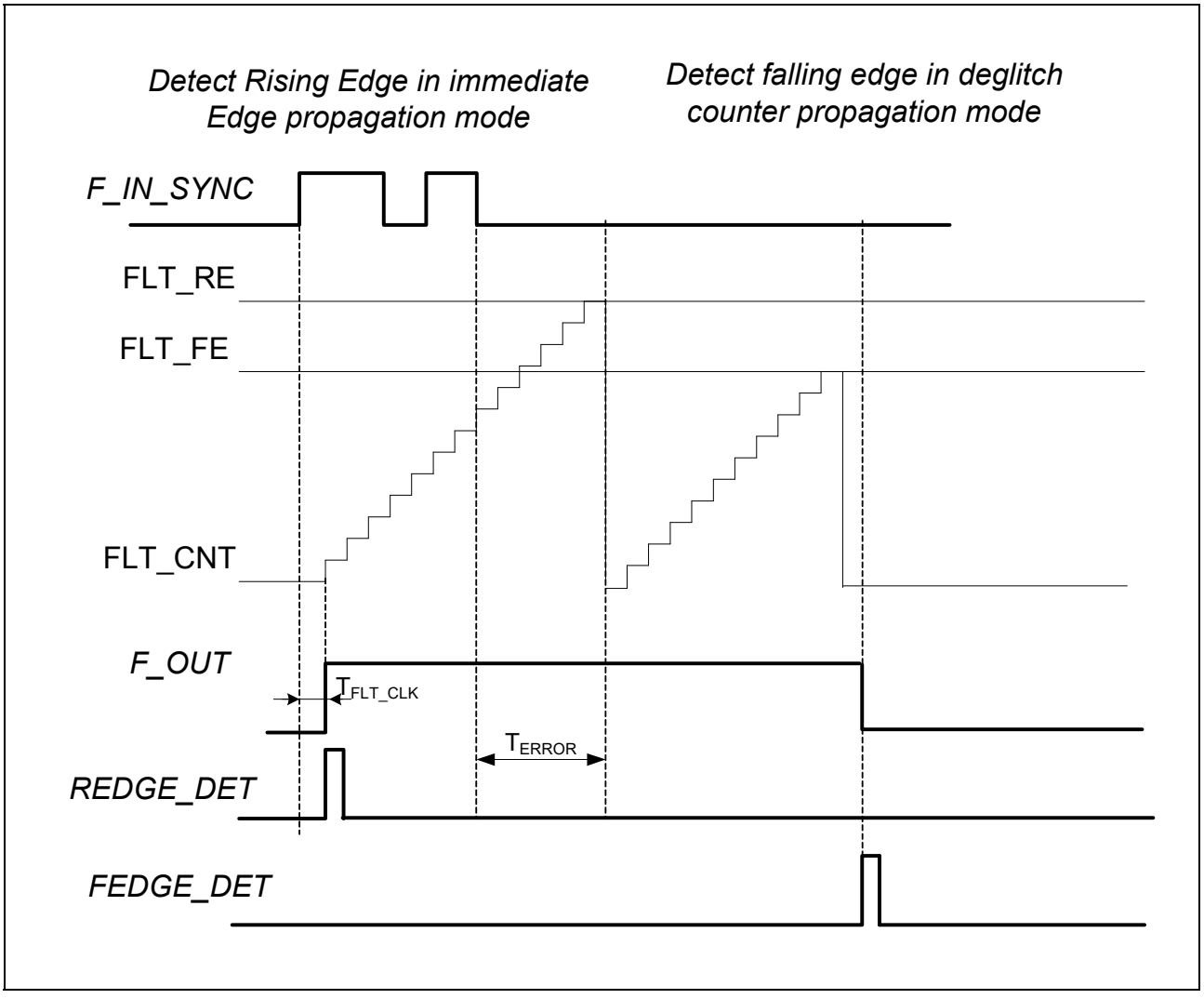



Figure 25.28 Mixed mode measurement

#### 25.10.2.3 TIM Filter reconfiguration

If FLT_EN=1 a change of FLT_RE or FLT_FE will take place immediately.

If FLT_EN=1 a change of FLT_MODE_RE or FLT_MODE_FE will be used with the next occurring corresponding edge. If the mode is changed while the filter unit is processing a certain mode, it will end this edge filtering in the mode as started.

If FLT_EN=1 a change of FLT_CTR_RE or FLT_CTR_FE will take place immediately.

### 25.10.3 Timeout Detection Unit (TDU)

The Timeout Detection Unit (TDU) is responsible for timeout detection of the TIM input signals.

Each channel of the TIM submodule has its own Timeout Detection Unit (TDU) where a timeout event can be set up on the filtered input signal of the corresponding channel.

The TDU architecture is shown in **Section 25.10.3.1, Architecture of the TDU Subunit**.

#### 25.10.3.1 Architecture of the TDU Subunit

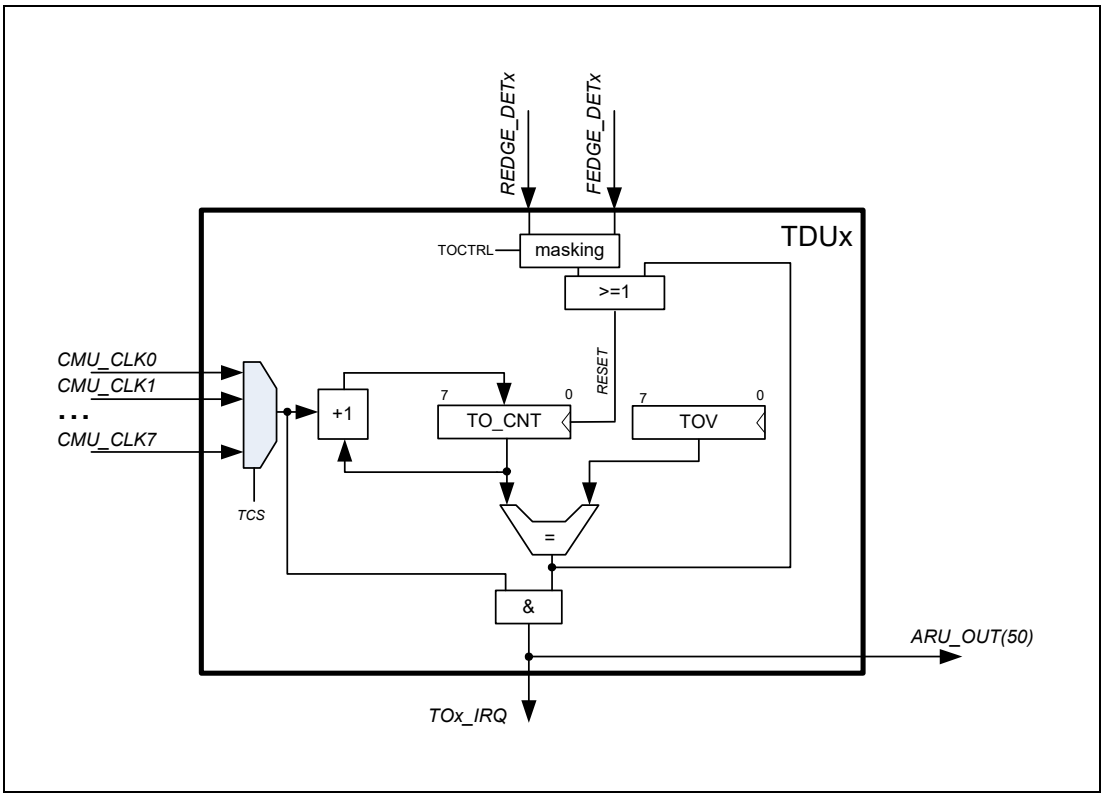



**Figure 25.29 Architecture of the TDU Subunit**

It is possible to detect timeouts with the resolution of the specified CMU_CLKx input signal selected with the bit field TCS of the register GTM0TIMixTDUV. The individual timeout values have to be specified in number of ticks of the selected input clock signal and have to be specified in the field TOV of timeout value register GTM0TIMixTDUV of the TIM channel x.

The exact time out value TTDU can be calculated with:

$T_{TDU}$=(TOV+1)*$T_{CMU_CLKx}$,

whereas TCMU_CLKx is the clock period of the selected CMU clock signal.

Timeout detection can be enabled or disabled individually inside the GTM0TIMixCTRLregister by setting/resetting the TOCTRL bit.

Timeout detection can be enabled to be sensitive to falling, rising or both edges of the input signal by writing the corresponding values to the bit field TOCTRL.

The counter TO_CNT is reset by each detected valid input edge coming either from the filtered input signal or when the timeout value TOV is reached by the counter TO_CNT.

After such a reset or by enabling the channel inside the GTM0TIMixCTRLregister the counter TO_CNT starts counting again from value 0 with the specified clock input signal.

Otherwise, timeout measurements starts immediately after the TOCTRL bit inside the GTM0TIMixCTRLregister is written (enabled).

The TDU generates an interrupt signal TIM_TODETx_IRQ whenever a timeout is detected for an individual input signal, and the TODET bit is set inside the GTM0TIMixIRQNOTIFY register.

In addition, when the ARU access is enabled with the ARU_EN bit inside the GTM0TIMixCTRLregister, the actual values stored inside the registers GTM0TIMixGPR0 and GTM0TIMixGPR1 are sent together with the last stored signal level to the ARU if a timeout event occurs.

To signal that a timeout occurred, the ARU_OUT(50) bit (ACB(2)) is set. The bit ACB(0) will be updated with the timeout event to the signal level on which the timeout was detected.

Thus, a destination could determine if a timeout occurred at the TIM input by evaluating ACB bit 2.

Since the TIM channel still monitors its input pin although the timeout happened, a valid edge could occur at the input pin while the timeout information is still valid at the ARU. In that case, the new edge associated data is stored inside the registers GTM0TIMixGPR0 and GTM0TIMixGPR1, the GPR overflow detected bit is set together in the ACB field (ACB(1)) with the timeout bit (ACB(2)) and the values are marked as valid to the ARU.

The ACB bit 2 is cleared, when a successful ARU write access by the TIM channel took place.

The ACB bit 1 is cleared, when a successful ARU write access by the TIM channel took place.

When a valid edge initiates an ARU write access which has not ended while a new timeout occurs the GPR overflow detected bit (ACB(1)) is set. The bit ACB(0) will be updated to the level on which the timeout occurred.

When a timeout occurred and initiates an ARU write access which has not ended while a new timeout occurs the GPR overflow detected bit (ACB(1)) is not set.

The following table clarifies the meaning of the ACB Bits for valid data provided by a TIM channel:

**Table 25.61 The ACB Bits for valid data provided by a TIM channel**

| ACB4/3 | ACB2 | ACB1 | ACB0 | Description |
|---|---|---|---|---|
| dc | 0 | 0 | SL | Valid edge detected |
| dc | 0 | 1 | SL | Input edge overwritten by subsequent edge |
| dc | 1 | 0 | SL | Timeout detected without valid edge |
| dc | 1 | 1 | SL | Timeout detected with subsequent valid edge detected |

## 25.10.4 TIM Channel Architecture

### 25.10.4.1 Overview

Each TIM channel consist of an input edge counter ECNT, a Signal Measurement Unit (SMU) with a counter CNT, a counter shadow register CNTS for SMU counter and two general purpose registers GPR0 and GPR1 for value storage.

The value TOV of the timeout register GTM0TIMixTDUV is provided to TDU subunit of each individual channel for timeout measurement. The architecture of the TIM channel is depicted in **Figure 25.30**.

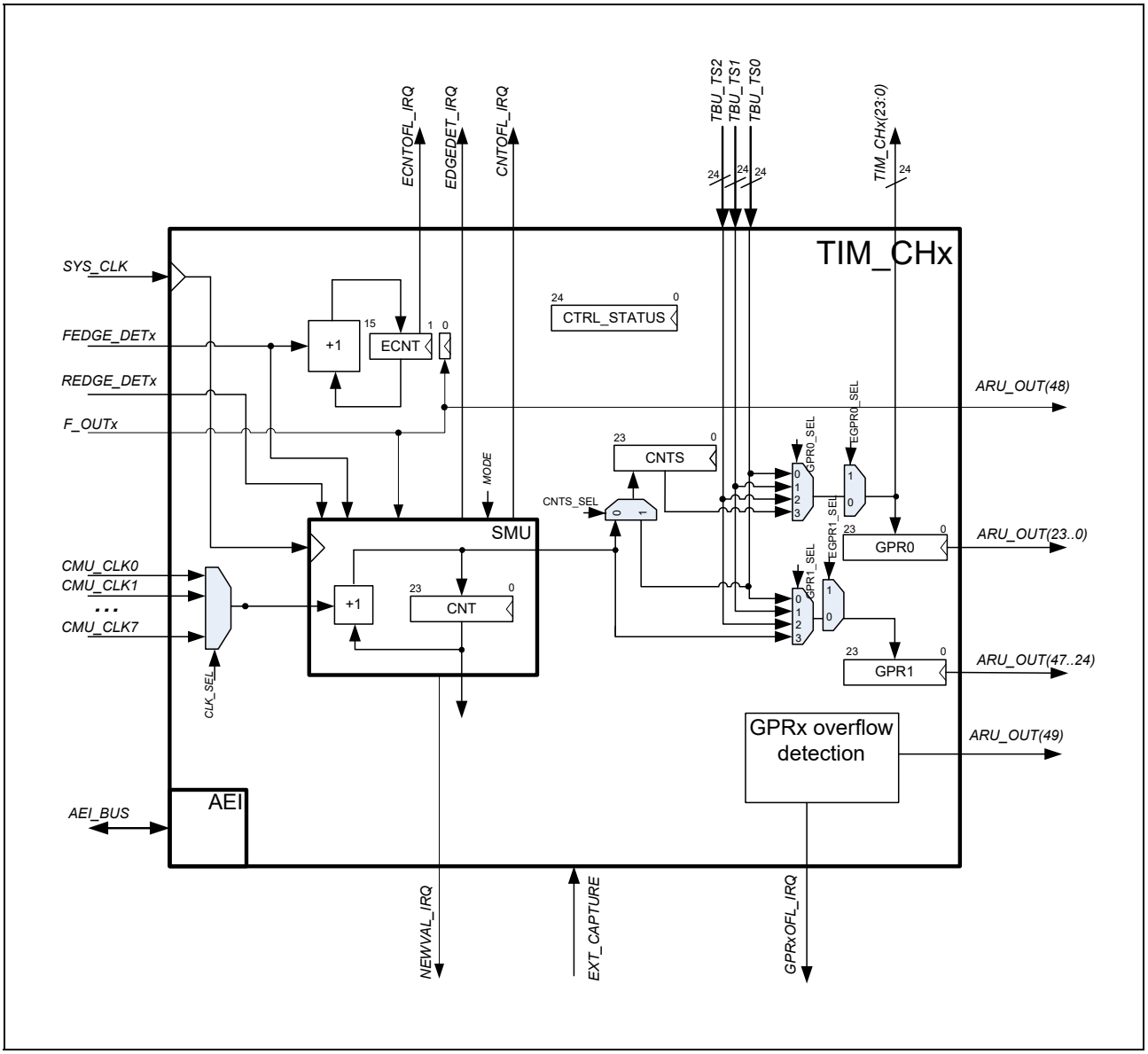



**Figure 25.30 TIM Channel Architecture**

Each TIM channel receives both input trigger signals REDGE_DETx and FEDGE_DETx, generated by the corresponding filter module in order to signalize a detected echo of the input signal F_INx. The signal F_OUTx shows the filtered signal of the channel's input signal F_INx.

The edge counter ECNT counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level

is part of the counter and can be obtained by bit 0 of ECNT. (However, the actual counter implementation counts only falling edges on ECNT[n:1] bits. It generates ECNT by composing the ECNT[n:1] bits with F_OUTx as bit 0).

Thus, the whole ECNT counter value is always odd, when a positive edge was received and always even, when a negative edge was received.

The current ECNT[7:0] register content is made visible on the bits 31 down to 24 of the registers GPR0, GPR1, and CNTS. This allows the software to detect inconsistent read accesses to registers GPR0, GPR1, and CNTS. However, the update strategy of these registers depends on the selected TIM modes, and thus the consistency check has to be adapted carefully.

It can be chosen with the bit field FR_ECNT_OFL when an ECNT overflow is signalled on ECNTOFL. An ECNT overflow can be signalled on 8 bit or full range resolution.

While reading the register GTM0TIMixECNT the bit ECNT[0] shows the input signal value F_OUTx independent of the state (enabled / disabled) of the channel. If a channel gets disabled (OSM mode or resetting TIM_EN) the content of GTM0TIMixECNT will be frozen until a read of the register takes place. This read will reset the ECNT counter. Continuing reads will show the input signal value in bit ECNT[0] again.

When new data is written into GPR0 and GPR1 the NEWVAL bit is set in GTM0TIMixIRQNOTIFY register and depending on corresponding enable bit value the NEWVALx_IRQ interrupt is raised.

Each TIM input channel has an ARU connection for providing data via the ARU to the other GTM submodules. The data provided to the ARU depends on the TIM channel mode and its corresponding adjustments (e.g. multiplexer configuration).

The bit ARU_EN of register GTM0TIMixCTRLdecides, whether the measurement results of registers GPR0 and GPR1 are consumed by another submodule via ARU (ARU_EN = 1) or the CPU via AEI (ARU_EN = 0).

To guarantee a consistent delivery of data from the GPR0 and GPR1 registers to the ARU or the CPU each TIM channel has to ensure that the data is consumed before it is overwritten with new values.

If new data was produced by the TIM channel (bit NEWVAL is set inside GTM0TIMixIRQNOTIFY register) while the old data is not consumed by the ARU (ARU_EN = 1) or CPU (ARU_EN = 0), the TIM channel sets the GPROFL bit inside the status register GTM0TIMixIRQNOTIFY and it overwrites the data inside the registers GPR0 and GPR1. In addition when ARU_EN = 1 the bit ACB(1) is set to 1 to indicate the overflow in the ARU data.

If the CPU is selected as consumer for the registers GPR0 and GPR1 (ARU_EN = 0), the acknowledge for reading out data is performed by a read access to the register GPR0. Thus, register GPR1 should be read always before GPR0.

If the ARU is selected as consumer for the registers GPR0 and GPR1 (ARU_EN = 1), the acknowledge for reading out data is performed by the ARU itself. However, the registers GPR0 and GPR1 could be read by CPU without giving an acknowledge.

### 25.10.4.2 TIM Channel Modes

The TIM provides six different measurement modes that can be configured with the bit field TIM_MODE of register GTM0TIMixCTRL. The measurement modes are described in the following subsections. Besides these different basic measurement modes, there exist distinct configuration bits in the register GTM0TIMixCTRLfor a more detailed controlling of each mode. The meanings of these bits are as follows:

- DSL: control the signal level for the measurement modes (e.g. if a measurement is started with rising edge or falling edge, or if high level pulses or low level pulses are measured.
- EGPR0_SEL, GPR0_SEL and EGPR1_SEL, GPR1_SEL: control the actual content of the registers GPR0 and GPR1 after a measurement has finished.
- CNTS_SEL: control the content of the registers CNTS. The actual time for updating the CNTS register is mode dependent.
- OSM: activate measurement in one-shot mode or continuous mode. In one-shot mode only one measurement cycle is performed and after that the channel is disabled.
- NEWVAL: The NEWVAL IRQ interrupt is triggered at the end of a measurement cycle, signalling that the registers GPR0 and GPR1 are updated.
- ARU_EN: enables sending of the registers GPR0 and GPR1 together with the actual signal level (in bit 48) and the overflow signal GPROFL (in bit 49), and the timeout status information (bit 50) to the ARU.
- EXT_CAP_EN: forces an update of the registers GPR0 and GPR1 and CNTS (TIM channel mode dependant) only on each rising edge of the EXT_CAPTURE signal and triggers a NEWVAL IRQ interrupt. If this mode is disabled the NEWVAL IRQ interrupt is triggered at the end of each measurement cycle.

For each channel the source of the EXT_CAPTURE signal can be configured with the bit fields EXT_CAP_SRC in the register GTM0TIM00ECTRL.

#### (1) TIM PWM Measurement Mode (TPWM)

In TIM PWM Measurement Mode the TIM channel measures duty cycle and period of an incoming PWM signal. The DSL bit defines the polarity of the PWM signal to be measured.

When measurement of pulse high time and period is requested (PWM with a high level duty cycle, DSL=1), the channel starts measuring after the first rising edge is detected by the filter.

Measurement is done with the CNT register counting with the configured clock coming from CMU_CLKx until a falling edge is detected.

Then the counter value is stored inside the shadow register CNTS (if CNTS_SEL = 0) and the counter CNT counts continuously until the next rising edge is reached.

On this following rising edge the content of the CNTS register is transferred to GPR0 and the content of CNT register is transferred to GPR1, assuming settings for the selectors GPR0_SEL= 11 and GPR1_SEL= 11. By this, GPR0 contains the duty cycle length and GPR1 contains the period. It should be noted, that the bits 1 to 7 of the ECNT may be used to check data consistency of the registers GPR0 and GPR1.

In addition the CNT register is cleared NEWVAL status bit inside of GTM0TIMixIRQNOTIFY status register and depending on corresponding interrupt enable condition TIM_NEWVALx_IRQ interrupt is raised.

The CNTS register update is not performed until the measurement is started (first edge defined by DSL is detected). Afterwards each edge leaving the level defined by DSL is performing a CNTS register update.

If a PWM with a low level duty cycle should be measured (DSL = 0), the channel waits for a falling edge until measurement is started. On this edge the low level duty cycle time is stored first in CNTS and then finally in GPR0 and the period is stored in GPR1.

When a PWM period was successfully measured, the data in the registers GPR0 and GPR1 is marked as valid for reading by the ARU when the ARU_EN bit is set inside GTM0TIMixCTRLregister, the NEWVAL bit is set inside the GTM0TIMixIRQNOTIFY register, and a new measurement is started.

If the preceding PWM values were not consumed by a reader attached to the ARU (ARU_EN bit enabled) or by the CPU the TIM channel set GPROFL status bit in GTM0TIMixIRQNOTIFY and depending on corresponding interrupt enable bit value raises a GPROFL_IRQ and overwrites the old values in GPR0 and GPR1. A new measurement is started afterwards.

If the register CNT produces an overflow during the measurement, the bit CNTOFL is set inside the register GTM0TIMixIRQNOTIFY and interrupt TIM_CNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

If the register ECNT produces an overflow during the measurement, the bit ECNTOFL is set inside the register GTM0TIMixIRQNOTIFY and interrupt TIM_ECNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

### (a) External capture TIM PWM Measurement Mode (TPWM)

If external capture is enabled, the pwm measurement is done continuously. The actual measurement values are captured to GPRx if an external capture event occurs.

Operation is done depending on cmu clock, ISL, DSL bit and the input signal value defined in next table:

**Table 25.62 External capture TIM PWM Measurement Mode (TPWM)**

| Input signal F_OUTx | selected CMU Clock | External capture | ISL | DSL | Action description |
|---|---|---|---|---|---|
| 0 | 1 | 0 | — | 0 | CNT++ |
| 1 | 1 | 0 | — | 0 | no |
| Rising edge | — | 0 | 0 | 0 | capture CNT value in CNTS |
| Falling edge | — | 0 | 0 | 0 | CNT = 0 |
| Rising edge | — | 0 | 1 | 0 | no |
| Falling edge | — | 0 | 1 | 0 | capture CNT value in CNTS; CNT = 0 |
| 1 | 1 | 0 | — | 1 | CNT++ |
| 0 | 1 | 0 | — | 1 | no |
| Falling edge | — | 0 | 0 | 1 | capture CNT value in CNTS |
| Rising edge | — | 0 | 0 | 1 | CNT = 0 |
| Falling edge | — | 0 | 1 | 1 | no |
| Rising edge | — | 0 | 1 | 1 | capture CNT value in CNTS; CNT = 0 |
| — | — | Rising edge | — | — | do GPRx capture ; issue NEWVAL_IRQ |
| — | 0 | 0 | — | — | no |

The CNTS register update is not performed until the measurement is started (first edge defined by DSL is detected). Afterwards the update of the CNTS register is defined by ISL,DSL combinations in the table above.

### (2) TIM Pulse Integration Mode (TPIM)

In TIM Pulse Integration Mode each TIM channel is able to measure a sum of pulse high or low times on an input signal, depending on the selected signal level bit DSL of register GTM0TIMixCTRLregister.

The pulse times are measured by incrementing the TIM channel counter CNT whenever the pulse has the specified signal level DSL. The counter is stopped whenever the input signal has the opposite signal level.

The counter CNT counts with the CMU_CLKx clock specified by the CLK_SEL bit field of the GTM0TIMixCTRLregister.

The CNT register is reset at the time the channel is activated (enabling via AEI write access) and it accumulates pulses while the channel is staying enabled.

Whenever the counter is stopped, the registers CNTS, GPR0 and GPR1 are updated according to settings of its corresponding input multiplexers, using the bits EGPR0_SEL, EGPR1_SEL, GPR0_SEL, GPR1_SEL, and CNTS_SEL. It should be noted, that the bits 1 to 7 of the ECNT may be used to check data consistency of the registers GPR0 and GPR1.

When the ARU_EN bit is set inside the GTM0TIMixCTRLregister the measurement results of the registers GPR0 and GPR1 can be send to subsequent submodules attached to the ARU.

#### (a) External capture TIM Pulse Integration Mode (TPIM)

If external capture is enabled, the pulse integration is done until next external capture event occurs.

Operation is done depending on cmu clock, DSL bit and the input signal value defined in next table:

**Table 25.63 External capture TIM Pulse Integration Mode (TPIM)**

| Input signal F_OUTx | selected CMU Clock | External capture | ISL | DSL | Action description |
|---|---|---|---|---|---|
| 0 | 1 | 0 | — | 0 | CNT++ |
| 1 | 1 | 0 | — | 0 | no |
| 1 | 1 | 0 | — | 1 | CNT++ |
| 0 | 1 | 0 | — | 1 | no |
| — | — | Rising edge | — | — | do capture ; issue NEWVAL_IRQ; CNT = 0 |
| — | 0 | 0 | — | — | no |

### (3) TIM Input Event Mode (TIEM)

In TIM Input Event Mode the TIM channel is able to count edges.

It is configurable if rising, falling or both edges should be counted. This can be done with the bit fields DSL and ISL in GTM0TIMixCTRLregister.

In addition, a TIM[i]_NEWVAL[x]_IRQ interrupt is raised when the configured edge was received and this interrupt was enabled.

The counter register CNT is used to count the number of edges, and the bit fields EGPR0_SEL, EGPR1_SEL,GPR0_SEL, GPR1_SEL, and CNTS_SEL can be used to configure the desired update values for the registers GPR0, GPR1 and CNTS. These register are updated whenever the edge counter CNT is incremented due to the arrival of a desired edge.

If the preceding data was not consumed by a reader attached to the ARU or by the CPU the TIM channel sets GPROFL status bit and raises a GPROFL[x]_IRQ if it was enabled in GTM0TIMixIRQEN register and overwrites the old values in GPR0 and GPR1 with the new ones.

If the register CNT produces an overflow during the measurement, the bit CNTOFL is set inside the register GTM0TIMixIRQNOTIFY and interrupt TIM_CNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

If the register ECNT produces an overflow during the measurement, the bit ECNTOFL is set inside the register GTM0TIMixIRQNOTIFY and interrupt TIM_ECNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

The TIM Input Event Mode does not depend on the bit field CLK_SEL of register GTM0TIMixCTRL.

#### (a) External capture TIM Input Event Mode (TIEM)

If external capture is enabled, capturing is done depending on the DSL, ISL bit and the input signal value defined in next table:

**Table 25.64 External capture TIM Input Event Mode (TIEM)**

| Input signal F_OUTx | External capture | ISL | DSL | Action description |
|---|---|---|---|---|
| — | Rising edge | 1 | — | do capture; issue NEWVAL_IRQ; CNT++ |
| — | 0 | 1 | — | no |
| 1 | Rising edge | 0 | 1 | do capture; issue NEWVAL_IRQ; CNT++ |
| 0 | — | 0 | 1 | no |
| 0 | Rising edge | 0 | 0 | do capture; issue NEWVAL_IRQ; CNT++ |
| 1 | — | 0 | 0 | no |

### (4) TIM Input Prescaler Mode (TIPM)

In the TIM Input Prescaler Mode the number of edges which should be detected before a TIM[i]_NEWVAL[x]_IRQ is raised is programmable. In this mode it must be specified in the CNTS register after how many edges the interrupt has to be raised.

A value of 0 in CNTS means that after one edge an interrupt is raised and a value of 1 means that after two edges an interrupt is raised, and so on.

The edges to be counted can be selected by the bit fields DSL and ISL of register GTM0TIMixCTRL.

With each triggered interrupt, the registers GPR0 and GPR1 are updated according to bits EGPR0_SEL, EGPR1_SEL,GPR0_SEL and GPR1_SEL.

If the register ECNT produces an overflow during the measurement, the bit ECNTOFL is set inside the register GTM0TIMixIRQNOTIFY and interrupt TIM_ECNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

The TIM Input Prescaler Mode does not depend on the bit field CLK_SEL of register GTM0TIMixCTRL.

#### (a) External capture TIM Input Prescaler Mode (TIPM)

If external capture is enabled, the external capture events are counted instead of the input signal edges.

Operation is done depending on the external capture signal, DSL, ISL bit and the input signal value defined in next table:

**Table 25.65 External capture TIM Input Prescaler Mode (TIPM)**

| Input signal F_OUTx | External capture | ISL | DSL | Action description |
|---|---|---|---|---|
| — | Rising edge | 1 | — | if CNT == CNTS then<br>do capture ;issue NEWVAL_IRQ; CNT = 0<br>else<br>CNT++<br>endif |
| — | 0 | 1 | — | no |
| 1 | Rising edge | 0 | 1 | if CNT == CNTS then<br>do capture ;issue NEWVAL_IRQ; CNT = 0<br>else<br>CNT++<br>endif |
| 0 | — | 0 | 1 | no |
| 0 | Rising edge | 0 | 0 | if CNT == CNTS then<br>do capture ;issue NEWVAL_IRQ; CNT = 0<br>else<br>CNT++<br>endif |
| 1 | — | 0 | 0 | no |

### (5) TIM Bit Compression Mode (TBCM)

The TIM Bit Compression Mode can be used to combine all filtered input signals of a TIM submodule to a parallel m bit data word, which can be routed to the ARU, where m is the number of channels available in the TIM submodule.

**Figure 25.31** gives an overview of the TIM bit compression mode.

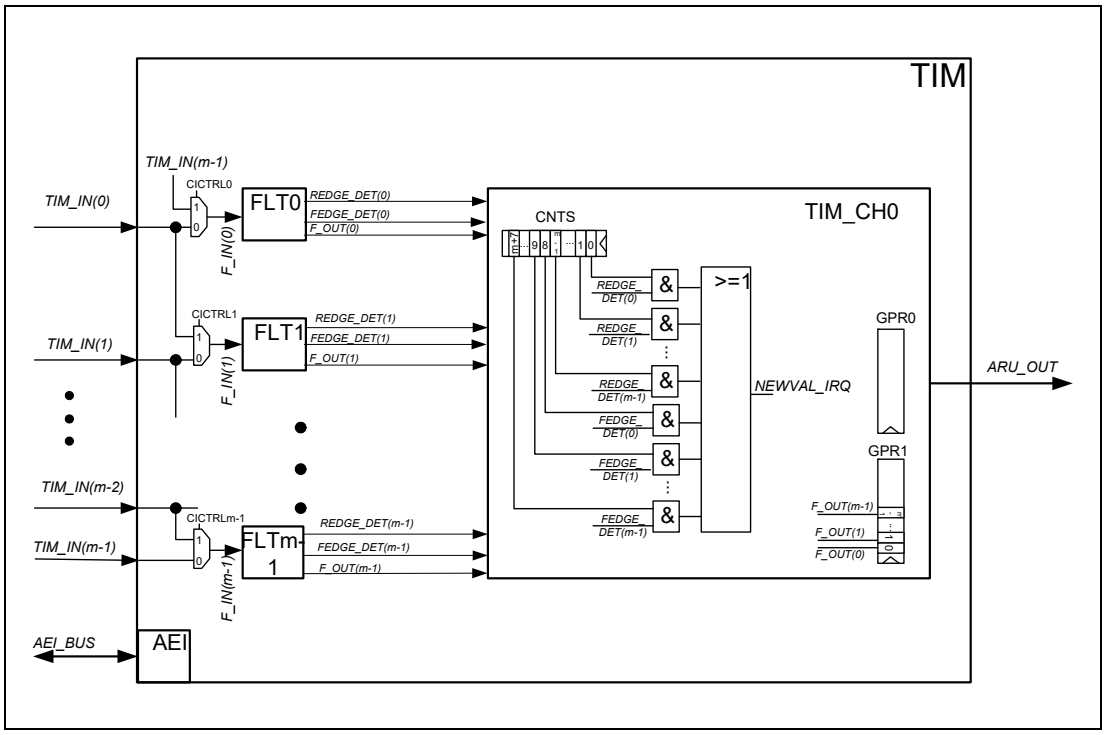



**Figure 25.31 TIM Bit Compression Mode**

The register CNTS of a channel is used to configure the event that releases the NEWVAL_IRQ and samples the input signals F_IN(0) to F_IN(m–1) in ascending order as a parallel data word in GPR1.

The bits 0 to m–1 of the CNTS register are used to select the REDGE_DET signals of the TIM filters 0 to m–1 as a sampling event, and the bits 8 to (7+m) are used to select the FEDGE_DET signals of the TIM filters 0 to m–1, respectively. If multiple events are selected, the events are OR-combined (see also **Figure 25.31**).

EGPR0_SEL,GPR0_SEL selects the timestamp value, which is routed through the ARU. GPR1_SEL is not applicable in TBCM mode.

If the bit ARU_EN of register GTM0TIMixCTRLis set, the sampled data of register GPR1 is routed together with a time stamp of register GPR0 to the ARU, whenever the NEWVAL_IRQ is released.

In TIM Bit compression mode, the register ECNT increments with each NEWVAL_IRQ, which means that the value of ECNT may depend on all m input signals. Consequently, the LSB of ECNT does not reflect the actual level of the input signal TIM_IN(x).

If the register ECNT produces an overflow during the measurement, the bit ECNTOFL is set inside the register GTM0TIMixIRQNOTIFY and interrupt TIM_ECNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

The TIM Bit Compression Mode does not depend on the bit field CLK_SEL of register GTM0TIMixCTRL.

(a) External capture Bit Compression Mode (TBCM)

If external capture is enabled, capturing is done depending on the DSL, ISL bit and the input signal value defined in next table:

**Table 25.66 External capture Bit Compression Mode (TBCM)**

| Input signal F_OUTx | External capture | ISL | DSL | Action description |
|---|---|---|---|---|
| — | Rising edge | 1 | — | do capture ;issue NEWVAL_IRQ; CNT++ |
| — | 0 | 1 | — | no |
| 1 | Rising edge | 0 | 1 | do capture ;issue NEWVAL_IRQ; CNT++ |
| 0 | — | 0 | 1 | no |
| 0 | Rising edge | 0 | 0 | do capture ;issue NEWVAL_IRQ; CNT++ |
| 1 | — | 0 | 0 | no |

### (6) TIM Gated Periodic Sampling Mode (TGPS)

In the TIM Gated Periodic Sampling Mode the number of CMU clock cycles which should elapse before capturing and raising TIM[i]_NEWVAL[x]_IRQ is programmable. In this mode it must be specified in the CNTS register after how many CMU clock cycles the interrupt has to be raised.

A value of 0 in GTM0TIMixCNTS means that after one CLK_SEL edge a trigger/interrupt is raised, and a value of 1 means that after two edges a trigger/interrupt is raised, and so on.

In the GTM0TIMixCNT register the elapsed cycles were incremented and compared against GTM0TIMixCNTS. If GTM0TIMixCNT is greater or equal to GTM0TIMixCNTS a trigger will be raised. This allows by writing a value to GTM0TIMixCNTS that the actual period time can be changed on the fly.

Operation is done depending on cmu clock, DSL, ISL bit and the input signal value defined in next table:

**Table 25.67 TIM Gated Periodic Sampling Mode (TGPS)**

| Input signal F_OUTx | selected CMU Clock | External capture | ISL | DSL | Action description |
|---|---|---|---|---|---|
| — | 1 | 0 | 1 | — | if CNT == CNTS then<br>do capture ;issue NEWVAL_IRQ; CNT = 0<br>else<br>CNT++<br>endif |
| 0 | 0 | 0 | 0 | 1 | no |
| 1 | 1 | 0 | 0 | 1 | if CNT == CNTS then<br>do capture ;issue NEWVAL_IRQ; CNT = 0<br>else<br>CNT++<br>endif |
| 0 | 0 | — | 0 | 1 | no |
| 0 | 1 | 0 | 0 | 0 | if CNT == CNTS then<br>do capture ;issue NEWVAL_IRQ; CNT = 0<br>else<br>CNT++<br>endif |
| 1 | 0 | 0 | 0 | 0 | no |
| — | 0 | 0 | — | — | no |

In this mode the GTM0TIMixGPR1 operates as a shadow register for GTM0TIMixCNTS. This would allow that the period for the next sampling period could be specified. The update of GTM0TIMixCNTS will only take place once on a trigger if the GTM0TIMixGPR1 was written by the CPU. This means that the captured value from the previous trigger can be read by the CPU from GTM0TIMixGPR1 and afterwards the new sampling period for the next sampling period (the one after the actual sampling period) could be written.

With each triggered interrupt, the registers GPR0 and GPR1 are updated according to bits GPR0_SEL, GPR1_SEL, EGPR0_SEL and EGPR1_SEL.

When selecting ECNT as a source for the capture registers, GPRx will show the edge count and the input signal value at point of capture. Selecting GPR0_SEL = '11' and EGPR0_SEL = '0' for TIM channel 0 all 8 TIM input signals will be captured to GPR0[7:0].

In the TGPS Mode the bit field CLK_SEL of register GTM0TIMixCTRLwill define the selected CMU clock which will be used.

The behavior of the ECNT counter is configurable by ECNT_RESET. If set to 1 on each interrupt (period expired) the ECNT will be reset. Otherwise it operates in wrap around mode.

If the register ECNT produces an overflow during the measurement, the bit ECNTOFL is set inside the register GTM0TIMixIRQNOTIFY and interrupt TIM_ECNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

### (a) External capture TIM Gated Periodic Sampling Mode (TGPS)

If external capture is enabled, the external capture events will capture the GPRx, reset the counter CNT and issue a NEWVAL_IRQ.

Operation is done depending on the cmu clock, external capture signal, DSL, ISL bit and the input signal value defined in next table:

**Table 25.68 External capture TIM Gated Periodic Sampling Mode (TGPS)**

| Input signal F_OUTx | selected CMU Clock | External capture | ISL | DSL | Action description |
|---|---|---|---|---|---|
| — | 1 | 0 | 1 | — | if CNT == CNTS then<br>do capture; issue NEWVAL_IRQ; CNT = 0<br>else<br>CNT++<br>endif |
| 0 | 0 | 0 | 0 | 1 | no |
| 1 | 1 | 0 | 0 | 1 | if CNT == CNTS then<br>do capture; issue NEWVAL_IRQ; CNT = 0<br>else<br>CNT++<br>endif |
| 0 | 0 | — | 0 | 1 | no |
| 0 | 1 | 0 | 0 | 0 | if CNT == CNTS then<br>do capture; issue NEWVAL_IRQ; CNT = 0<br>else<br>CNT++<br>endif |
| 1 | 0 | 0 | 0 | 0 | no |
| — | 0 | 0 | — | — | no |
| — | — | rising edge | — | — | do capture; issue NEWVAL_IRQ; CNT = 0 |

### 25.10.5 TIM Interrupt Signals

TIM provides 6 interrupt lines per channel. These interrupts are shown below:

**Table 25.69 TIM Interrupt Signals**

| Signal | Description |
|---|---|
| TIM[i]_NEWVAL[x]_IRQ | New measurement value detected by SMU of channel x (x: 0 to 7) |
| TIM[i]_ECNTOFL[x]_IRQ | ECNT counter overflow of channel x (x: 0 to 7) |
| TIM[i]_CNTOFL[x]_IRQ | SMU CNT counter overflow of channel x (x: 0 to 7) |
| TIM[i]_GPROFL[x]_IRQ | GPR0 and GPR1 data overflow, old data was not read out before new data has arrived at input pin of channel x (x: 0 to 7) |
| TIM[i]_TODET[x]_IRQ | Time out reached for input signal of channel x (x: 0 to 7) |
| TIM[i]_GLITCHDET_IRQ | A glitch was detected by the TIM filter of channel x (x: 0 to 7) |

### 25.10.6 TIM Configuration Registers Overview

TIM contains following configuration registers:

**Table 25.70 Register list**

| Symbol | Register Name | Detail in Section |
|---|---|---|
| GTM0TIMixCTRL | channel x control | **25.10.7.1**, **25.10.7.2** |
| GTM0TIM00ECTRL | channel x (x: 0 to 7) extended control | **25.10.7.19** |
| GTM0TIMixFLTRE | channel x (x: 0 to 7) filter parameter 0 | **25.10.7.3** |
| GTM0TIMixFLTFE | channel x (x: 0 to 7) filter parameter 1 | **25.10.7.4** |
| GTM0TIMixTDUV | channel x (x: 0 to 7) TDU control. | **25.10.7.16** |
| GTM0TIMixTDUC | channel x (x: 0 to 7) TDU counter. | **25.10.7.17** |
| GTM0TIMixGPR0 | channel x (x: 0 to 7) general purpose 0 | **25.10.7.5** |
| GTM0TIMixGPR1 | channel x (x: 0 to 7) general purpose 1 | **25.10.7.6** |
| GTM0TIMixCNT | channel x (x: 0 to 7) SMU counter | **25.10.7.7** |
| GTM0TIMixECNT | channel x (x: 0 to 7) SMU edge counter | **25.10.7.18** |
| GTM0TIMixCNTS | channel x (x: 0 to 7) SMU shadow counter | **25.10.7.8** |
| GTM0TIMixIRQNOTIFY | channel x (x: 0 to 7) interrupt notification | **25.10.7.9** |
| GTM0TIMixIRQEN | channel x (x: 0 to 7) interrupt enable | **25.10.7.10** |
| GTM0TIMixEIRQEN | channel x (x: 0 to 7) error interrupt enable | **25.10.7.15** |
| GTM0TIMixIRQFORCINT | channel x (x: 0 to 7) software interrupt force | **25.10.7.11** |
| GTM0TIMixIRQMODE | IRQ mode configuration register (x: 0 to 7) | **25.10.7.12** |
| GTM0TIMiRST | TIM global software reset | **25.10.7.13** |
| GTM0TIMiINSRC | TIM AUX IN source selection | **25.10.7.14** |
| GTM0TIMiINPVAL | TIM input value observation | **25.10.7.20** |

## 25.10.7 TIM Configuration Registers Description

### 25.10.7.1 GTM0TIM1xCTRL (x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0TIM10ECTRL: <GTM_base> + $01824_H$
GTM0TIM11ECTRL: <GTM_base> + $018A4_H$
GTM0TIM12ECTRL: <GTM_base> + $01924_H$
GTM0TIM13ECTRL: <GTM_base> + $019A4_H$
GTM0TIM14ECTRL: <GTM_base> + $01A24_H$
GTM0TIM15ECTRL: <GTM_base> + $01AA4_H$
GTM0TIM16ECTRL: <GTM_base> + $01B24_H$
GTM0TIM17ECTRL: <GTM_base> + $01BA4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TOCTRL | | EGPR1_SEL | EGPR0_SEL | FR_ECNT_OFL | CLK_SEL | | | FLT_CTR_FE | FLT_MODE_FE | FLT_CTR_RE | FLT_MODE_RE | EXT_CAP_EN | FLT_CNT_FRQ | | FLT_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECNT_RESET | ISL | DSL | CNTS_SEL | GPR1_SEL | | GPR0_SEL | | — | CICTRL | ARU_EN | OSM | TIM_MODE | | | TIM_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.71 GTM0TIM1xCTRL Register Contents (1/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | TOCTRL | Timeout control<br>00: Timeout feature disabled<br>11: Timeout feature enabled for both edges<br>01: Timeout feature enabled for rising edge only<br>10: Timeout feature enabled for falling edge only |
| 29 | EGPR1_SEL | Extension of GPR1_SEL bit field.<br>Details described in GPR1_SEL bit field. |
| 28 | EGPR0_SEL | Extension of GPR0_SEL bit field.<br>Details described in GPR0_SEL bit field. |
| 27 | FR_ECNT_OFL | Extended Edge counter overflow behavior<br>0: Overflow will be signalled on ECNT bit width = 8<br>1: Overflow will be signalled on EECNT bit width (full range) |
| 26 to 24 | CLK_SEL | CMU clock source select for channel.<br>000: CMU_CLK0 selected<br>001: CMU_CLK1 selected<br>010: CMU_CLK2 selected<br>011: CMU_CLK3 selected<br>100: CMU_CLK4 selected<br>101: CMU_CLK5 selected<br>110: CMU_CLK6 selected<br>111: CMU_CLK7 selected |
| 23 | FLT_CTR_FE | Filter counter mode for falling edge.<br>0: Up/Down Counter<br>1: Hold Counter<br>**NOTE**<br>This bit is only applicable in Individual Deglitch Time Mode. |

Table 25.71 GTM0TIM1xCTRL Register Contents (2/4)

| Bit Position | Bit Name | Function |
|---|---|---|
| 22 | FLT_MODE_FE | Filter mode for falling edge.<br>0: Immediate edge propagation mode<br>1: Individual de-glitch mode |
| 21 | FLT_CTR_RE | Filter counter mode for rising edge.<br>0: Up/Down Counter<br>1: Hold Counter<br>**NOTE**<br>This bit is only applicable in Individual Deglitch Time Mode. |
| 20 | FLT_MODE_RE | Filter mode for rising edge.<br>0: Immediate edge propagation mode<br>1: Individual de-glitch mode |
| 19 | EXT_CAP_EN | Enables external capture mode. The selected TIM mode is only sensitive to external capture pulses the input event changes are ignored.<br>0: External capture disabled<br>1: External capture enabled |
| 18, 17 | FLT_CNT_FRQ | Filter counter frequency select<br>00: FLT_CNT counts with CMU_CLK0<br>01: FLT_CNT counts with CMU_CLK1<br>10: FLT_CNT counts with CMU_CLK6<br>11: FLT_CNT counts with CMU_CLK7 |
| 16 | FLT_EN | Filter enable for channel x (x = 0 to 7)<br>0: Filter disabled and internal states are reset<br>1: Filter enabled<br>**NOTE**<br>If the filter is disabled all filter related units (including CSU) are bypassed, which means that the signal F_IN is directly routed to signal F_OUT. |
| 15 | ECNT_RESET | Enables resetting the ECNT counter in periodic sampling mode<br>0: ECNT counter operating in wrap around mode<br>1: ECNT counter is reset with periodic sampling |
| 14 | ISL | Ignore signal level<br>0: Use DSL bit for selecting active signal level<br>1: Ignore DSL and treat both edges as active edge<br>**NOTE**<br>This bit is only applicable in Input Event mode (TIEM). |
| 13 | DSL | Signal level control<br>0: Measurement starts with falling edge (low level measurement)<br>1: Measurement starts with rising edge (high level measurement) |
| 12 | CNTS_SEL | Selection for CNTS register<br>0: Use CNT register as input<br>1: Use TBU_TS0 as input<br>**NOTE**<br>The functionality of the CNTS_SEL is disabled in the modes TIPM and TBCM. |

**Table 25.71 GTM0TIM1xCTRL Register Contents (3/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11, 10 | GPR1_SEL | Selection for GPR1 register<br>If EGPR1_SEL =0:<br>00: Use TBU_TS0 as input<br>01: Use TBU_TS1 as input<br>10: Use TBU_TS2 as input<br>11: Use CNT as input<br>If EGPR1_SEL =1:<br>00: Use ECNT as input<br>01: Use TIM_INP_VAL as input<br>10: Reserved<br>11: Reserved<br>**NOTES**<br>1. In TBCM mode: EGPR1_SEL = 1, GPR1_SEL=01 selects TIM_INP_VAL as input, in all other cases TIM Filter F_OUT is used.<br>2. If a reserved value is written to the EGPR1_SEL, GPR1_SEL bit fields, the hardware will use TBU_TS0 input. |
| 9, 8 | GPR0_SEL | Selection for GPR0 register<br>If EGPR0_SEL =0:<br>00: Use TBU_TS0 as input<br>01: Use TBU_TS1 as input<br>10: Use TBU_TS2 as input<br>11: Use CNTS as input;if TGPS mode in channel = 0 is selected use TIM Filter F_OUT as input<br>If EGPR0_SEL =1:<br>00: Use ECNT as input<br>01: Use TIM_INP_VAL as input<br>10: Reserved<br>11: Reserved<br>**NOTE**<br>If a reserved value is written to the EGPR0_SEL, GPR0_SEL bit fields, the hardware will use TBU_TS0 input. |
| 7 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 6 | CICTRL | Channel Input Control.<br>0: Use signal TIM_IN(x) as input for channel x<br>1: Use signal TIM_IN(x-1) as input for channel x (or TIM_IN(m-1) if x is 0) |
| 5 | ARU_EN | GPR0 and GPR1 register values routed to ARU<br>0: Registers content not routed<br>1: Registers content routed |
| 4 | OSM | One-shot mode<br>0: Continuous operation mode<br>1: One-shot mode<br>**NOTE**<br>After finishing the action in one-shot mode the TIM_EN bit is cleared automatically. |

**Table 25.71 GTM0TIM1xCTRL Register Contents (4/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 to 1 | TIM_MODE | TIM channel x (x = 0 to 7) mode<br>000: PWM Measurement Mode (TPWM)<br>001: Pulse Integration Mode (TPIM)<br>010: Input Event Mode (TIEM)<br>011: Input Prescaler Mode (TIPM)<br>100: Bit Compression Mode (TBCM)<br>101: Gated Periodic Sampling Mode (TGPS)<br>**NOTES**<br>1. If an undefined value is written to the TIM_MODE register, the hardware switches automatically to TIM_MODE = 000 (TPWM mode).<br>2. The TIM_MODE register should not be changed while the TIM channel is enabled.<br>3. If the TIM channel is enabled and operating in TPWM or TPIM mode after the first valid edge defined by DSL has occurred, a reconfiguration of DSL,ISL,TIM_MODE will not change the channel behavior. Reading these bit fields after reconfiguration will show the newly configured settings but the initial channel behavior will not change. Only a disabling of the TIM channel by setting TIM_EN = 0 and reenabling with TIM_EN = 1 will change the channel operation mode. |
| 0 | TIM_EN | TIM channel x (x = 0 to 7) enable<br>0: Channel disabled<br>1: Channel enabled<br>**NOTES**<br>1. Enabling of the channel resets the registers ECNT, GTM0TIMixCNT, GTM0TIMixGPR0, and GTM0TIMixGPR1 to their reset values.<br>2. After finishing the action in one-shot mode the TIM_EN bit is cleared automatically. Otherwise, the bit must be cleared manually. |

### 25.10.7.2 GTM0TIM0xCTRL (x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0TIM00ECTRL: <GTM_base> + $01024_H$
GTM0TIM01ECTRL: <GTM_base> + $010A4_H$
GTM0TIM02ECTRL: <GTM_base> + $01124_H$
GTM0TIM03ECTRL: <GTM_base> + $011A4_H$
GTM0TIM04ECTRL: <GTM_base> + $01224_H$
GTM0TIM05ECTRL: <GTM_base> + $012A4_H$
GTM0TIM06ECTRL: <GTM_base> + $01324_H$
GTM0TIM07ECTRL: <GTM_base> + $013A4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TOCTRL | | EGPR1_SEL | EGPR0_SEL | FR_ECNT_OFL | CLK_SEL | | | FLT_CTR_FE | FLT_MODE_FE | FLT_CTR_RE | FLT_MODE_RE | EXT_CAP_EN | FLT_CNT_FRQ | | FLT_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECNT_RESET | ISL | DSL | CNTS_SEL | GPR1_SEL | | GPR0_SEL | | TBU0_SEL | CICTRL | ARU_EN | OSM | TIM_MODE | | | TIM_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.72 GTM0TIM0xCTRL Register Contents (1/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | TOCTRL | Timeout control<br>00: Timeout feature disabled<br>01: Timeout feature enabled for rising edge only<br>10: Timeout feature enabled for falling edge only<br>11: Timeout feature enabled for both edges |
| 29 | EGPR1_SEL | Extension of GPR1_SEL bit field.<br>Details described in GPR1_SEL bit field. |
| 28 | EGPR0_SEL | Extension of GPR0_SEL bit field.<br>Details described in GPR0_SEL bit field. |
| 27 | FR_ECNT_OFL | Extended Edge counter overflow behavior<br>0: Overflow will be signalled on ECNT bit width = 8<br>1: Overflow will be signalled on EECNT bit width (full range) |
| 26 to 24 | CLK_SEL | CMU clock source select for channel.<br>000: CMU_CLK0 selected<br>001: CMU_CLK1 selected<br>010: CMU_CLK2 selected<br>011: CMU_CLK3 selected<br>100: CMU_CLK4 selected<br>101: CMU_CLK5 selected<br>110: CMU_CLK6 selected<br>111: CMU_CLK7 selected |
| 23 | FLT_CTR_FE | Filter counter mode for falling edge.<br>0: Up/Down Counter<br>1: Hold Counter<br>**NOTE**<br>This bit is only applicable in Individual Deglitch Time Mode. |
| 22 | FLT_MODE_FE | Filter mode for falling edge.<br>0: Immediate edge propagation mode<br>1: Individual de-glitch mode |

**Table 25.72 GTM0TIM0xCTRL Register Contents (2/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 21 | FLT_CTR_RE | Filter counter mode for rising edge.<br>0: Up/Down Counter<br>1: Hold Counter<br>**NOTE**<br>This bit is only applicable in Individual Deglitch Time Mode. |
| 20 | FLT_MODE_RE | Filter mode for rising edge.<br>0: Immediate edge propagation mode<br>1: Individual de-glitch mode |
| 19 | EXT_CAP_EN | Enables external capture mode. The selected TIM mode is only sensitive to external capture pulses the input event changes are ignored.<br>0: External capture disabled<br>1: External capture enabled |
| 18, 17 | FLT_CNT_FRQ | Filter counter frequency select<br>00: FLT_CNT counts with CMU_CLK0<br>01: FLT_CNT counts with CMU_CLK1<br>10: FLT_CNT counts with CMU_CLK6<br>11: FLT_CNT counts with CMU_CLK7 |
| 16 | FLT_EN | Filter enable for channel x (x = 0 to 7)<br>0: Filter disabled and internal states are reset<br>1: Filter enabled<br>**NOTE**<br>If the filter is disabled all filter related units (including CSU) are bypassed, which means that the signal F_IN is directly routed to signal F_OUT. |
| 15 | ECNT_RESET | Enables resetting the ECNT counter in periodic sampling mode<br>0: ECNT counter operating in wrap around mode<br>1: ECNT counter is reset with periodic sampling |
| 14 | ISL | Ignore signal level<br>0: Use DSL bit for selecting active signal level<br>1: Ignore DSL and treat both edges as active edge<br>**NOTE**<br>This bit is only applicable in Input Event mode (TIEM and TIPM) |
| 13 | DSL | Signal level control<br>0: Measurement starts with falling edge (low level measurement)<br>1: Measurement starts with rising edge (high level measurement) |
| 12 | CNTS_SEL | Selection for CNTS register<br>0: Use CNT register as input<br>1: Use TBU_TS0 as input<br>**NOTE**<br>The functionality of the CNTS_SEL is disabled in the modes TIPM and TBCM. |

**Table 25.72 GTM0TIM0xCTRL Register Contents (3/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11, 10 | GPR1_SEL | Selection for GPR1 register<br>If EGPR1_SEL = 0:<br>00: Use TBU_TS0 as input<br>01: Use TBU_TS1 as input<br>10: Use TBU_TS2 as input<br>11: Use CNT as input<br>If EGPR1_SEL =1:<br>00: Use ECNT as input<br>01: Use TIM_INP_VAL as input<br>10: Reserved<br>11: Reserved<br>**NOTES**<br>1. In TBCM mode: EGPR1_SEL=1, GPR1_SEL=01 selects TIM_INP_VAL as input, in all other cases TIM Filter F_OUT is used.<br>2. If a reserved value is written to the EGPR1_SEL, GPR1_SEL bit fields , the hardware will use TBU_TS0 input. |
| 9, 8 | GPR0_SEL | Selection for GPR0 register<br>If EGPR0_SEL = 0:<br>00: Use TBU_TS0 as input<br>01: Use TBU_TS1 as input<br>10: Use TBU_TS2 as input<br>11: Use CNTS as input;if TGPS mode in channel = 0 is selected use TIM Filter F_OUT as input<br>If EGPR0_SEL =1:<br>00: Use ECNT as input<br>01: Use TIM_INP_VAL as input<br>10: Reserved<br>11: Reserved<br>**NOTE**<br>If a reserved value is written to the EGPR0_SEL, GPR0_SEL bit fields, the hardware will use TBU_TS0 input. |
| 7 | TBU0_SEL | TBU_TS0 bits input select for TIM0_CH[x]_GPRz (z: 0, 1)<br>0: Use TBU_TS0(23 to 0) to store in TIM0_CH[x]_GPR0/TIM0_CH[x]_GPR1<br>1: Use TBU_TS0(26 to 3) to store in TIM0_CH[x]_GPR0/TIM0_CH[x]_GPR1 |
| 6 | CICTRL | Channel Input Control.<br>0: Use signal TIM_IN(x) as input for channel x<br>1: Use signal TIM_IN(x-1) as input for channel x (or TIM_IN(m-1) if x is 0) |
| 5 | ARU_EN | GPR0 and GPR1 register values routed to ARU<br>0: Registers content not routed<br>1: Registers content routed |
| 4 | OSM | One-shot mode<br>0: Continuous operation mode<br>1: One-shot mode<br>**NOTE**<br>After finishing the action in one-shot mode the TIM_EN bit is cleared automatically. |

**Table 25.72 GTM0TIM0xCTRL Register Contents (4/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 to 1 | TIM_MODE | TIM channel x (x = 0 to 7) mode<br>000: PWM Measurement Mode (TPWM)<br>001: Pulse Integration Mode (TPIM)<br>010: Input Event Mode (TIEM)<br>011: Input Prescaler Mode (TIPM)<br>100: Bit Compression Mode (TBCM)<br>101: Gated Periodic Sampling Mode (TGPS)<br>**NOTES**<br>1. If an undefined value is written to the TIM_MODE register, the hardware switches automatically to TIM_MODE = 000 (TPWM mode).<br>2. The TIM_MODE register should not be changed while the TIM channel is enabled.<br>3. If the TIM channel is enabled and operating in TPWM or TPIM mode after the first valid edge defined by DSL has occurred, a reconfiguration of DSL,ISL,TIM_MODE will not change the channel behavior. Reading these bit fields after reconfiguration will show the newly configured settings but the initial channel behavior will not change. Only a disabling of the TIM channel by setting TIM_EN = 0 and reenabling with TIM_EN = 1 will change the channel operation mode. |
| 0 | TIM_EN | TIM channel x (x = 0 to 7) enable<br>0: Channel disabled<br>1: Channel enabled<br>**NOTES**<br>1. Enabling of the channel resets the registers ECNT, GTM0TIMixCNT, GTM0TIMixGPR0, and GTM0TIMixGPR1 to their reset values.<br>2. After finishing the action in one-shot mode the TIM_EN bit is cleared automatically. Otherwise, the bit must be cleared manually. |

### 25.10.7.3 GTM0TIMixFLTRE (i = 0,1, x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0TIM00FLTRE: <GTM_base> + $0101C_H$, GTM0TIM10FLTRE: <GTM_base> + $0181C_H$
GTM0TIM01FLTRE: <GTM_base> + $0109C_H$, GTM0TIM11FLTRE: <GTM_base> + $0189C_H$
GTM0TIM02FLTRE: <GTM_base> + $0111C_H$, GTM0TIM12FLTRE: <GTM_base> + $0191C_H$
GTM0TIM03FLTRE: <GTM_base> + $0119C_H$, GTM0TIM13FLTRE: <GTM_base> + $0199C_H$
GTM0TIM04FLTRE: <GTM_base> + $0121C_H$, GTM0TIM14FLTRE: <GTM_base> + $01A1C_H$
GTM0TIM05FLTRE: <GTM_base> + $0129C_H$, GTM0TIM15FLTRE: <GTM_base> + $01A9C_H$
GTM0TIM06FLTRE: <GTM_base> + $0131C_H$, GTM0TIM16FLTRE: <GTM_base> + $01B1C_H$
GTM0TIM07FLTRE: <GTM_base> + $0139C_H$, GTM0TIM17FLTRE: <GTM_base> + $01B9C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | FLT_RE | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLT_RE | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.73 GTM0TIMixFLTRE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | FLT_RE | Filter parameter for rising edge.<br>**NOTE**<br>This register has different meanings in the various filter modes.<br>Immediate edge propagation mode = acceptance time for rising edge<br>Individual deglitch time mode = deglitch time for rising edge |

### 25.10.7.4 GTM0TIMixFLTFE (i = 0,1, x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0TIM00FLTFE: <GTM_base> + $01020_H$, GTM0TIM10FLTFE: <GTM_base> + $01820_H$
GTM0TIM01FLTFE: <GTM_base> + $010A0_H$, GTM0TIM11FLTFE: <GTM_base> + $018A0_H$
GTM0TIM02FLTFE: <GTM_base> + $01120_H$, GTM0TIM12FLTFE: <GTM_base> + $01920_H$
GTM0TIM03FLTFE: <GTM_base> + $011A0_H$, GTM0TIM13FLTFE: <GTM_base> + $019A0_H$
GTM0TIM04FLTFE: <GTM_base> + $01220_H$, GTM0TIM14FLTFE: <GTM_base> + $01A20_H$
GTM0TIM05FLTFE: <GTM_base> + $012A0_H$, GTM0TIM15FLTFE: <GTM_base> + $01AA0_H$
GTM0TIM06FLTFE: <GTM_base> + $01320_H$, GTM0TIM16FLTFE: <GTM_base> + $01B20_H$
GTM0TIM07FLTFE: <GTM_base> + $013A0_H$, GTM0TIM17FLTFE: <GTM_base> + $01BA0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | FLT_FE | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLT_FE | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.74 GTM0TIMixFLTFE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | FLT_FE | Filter parameter for falling edge.<br>**NOTE**<br>This register has different meanings in the various filter modes.<br>Immediate edge propagation mode = acceptance time for falling edge<br>Individual deglitch time mode = deglitch time for falling edge |

### 25.10.7.5 GTM0TIMixGPR0 (i = 0,1, x = 0 to 7)

**Access:** This register can be read in 32-bit units.

**Address:**

| | |
|---|---|
| GTM0TIM00GPR0: <GTM_base> + $01000_H$, | GTM0TIM10GPR0: <GTM_base> + $01800_H$ |
| GTM0TIM01GPR0: <GTM_base> + $01080_H$, | GTM0TIM11GPR0: <GTM_base> + $01880_H$ |
| GTM0TIM02GPR0: <GTM_base> + $01100_H$, | GTM0TIM12GPR0: <GTM_base> + $01900_H$ |
| GTM0TIM03GPR0: <GTM_base> + $01180_H$, | GTM0TIM13GPR0: <GTM_base> + $01980_H$ |
| GTM0TIM04GPR0: <GTM_base> + $01200_H$, | GTM0TIM14GPR0: <GTM_base> + $01A00_H$ |
| GTM0TIM05GPR0: <GTM_base> + $01280_H$, | GTM0TIM15GPR0: <GTM_base> + $01A80_H$ |
| GTM0TIM06GPR0: <GTM_base> + $01300_H$, | GTM0TIM16GPR0: <GTM_base> + $01B00_H$ |
| GTM0TIM07GPR0: <GTM_base> + $01380_H$, | GTM0TIM17GPR0: <GTM_base> + $01B80_H$ |

**Value after reset:** $0X00\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECNT | | | | | | | | GPR0 | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GPR0 | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.75 GTM0TIMixGPR0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | ECNT | Edge counter.<br>**NOTE**<br>The ECNT counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of ECNT. |
| 23 to 0 | GPR0 | Input signal characteristic parameter 0. |

**NOTES**

1. The ECNT register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.
2. The content of this register has different meaning for the TIM channels modes. The content directly depends on the bit fields EGPR0_SEL, GPR0_SEL of register GTM0TIMixCTRL.

### 25.10.7.6 GTM0TIMixGPR1 (i = 0,1, x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:**

| | |
|---|---|
| GTM0TIM00GPR1: <GTM_base> + $01004_H$, | GTM0TIM10GPR1: <GTM_base> + $01804_H$ |
| GTM0TIM01GPR1: <GTM_base> + $01084_H$, | GTM0TIM11GPR1: <GTM_base> + $01884_H$ |
| GTM0TIM02GPR1: <GTM_base> + $01104_H$, | GTM0TIM12GPR1: <GTM_base> + $01904_H$ |
| GTM0TIM03GPR1: <GTM_base> + $01184_H$, | GTM0TIM13GPR1: <GTM_base> + $01984_H$ |
| GTM0TIM04GPR1: <GTM_base> + $01204_H$, | GTM0TIM14GPR1: <GTM_base> + $01A04_H$ |
| GTM0TIM05GPR1: <GTM_base> + $01284_H$, | GTM0TIM15GPR1: <GTM_base> + $01A84_H$ |
| GTM0TIM06GPR1: <GTM_base> + $01304_H$, | GTM0TIM16GPR1: <GTM_base> + $01B04_H$ |
| GTM0TIM07GPR1: <GTM_base> + $01384_H$, | GTM0TIM17GPR1: <GTM_base> + $01B84_H$ |

**Value after reset:** $0X00\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECNT | | | | | | | | GPR1 | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | — | — | — | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GPR1 | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.76 GTM0TIMixGPR1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | ECNT | Edge counter.<br>**NOTE**<br>The ECNT counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of ECNT. |
| 23 to 0 | GPR1 | Input signal characteristic parameter 1.<br>**NOTE**<br>In TBCM mode if EGPR1_SEL=1, GPR1_SEL=01 then TIM_INP_VAL is used as input in all other cases TIM Filter F_OUT is used as input and Bits GPR1(23:8) = 0 |

**NOTES**

1. The ECNT register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.
2. The content of this register has different meaning for the TIM channels modes. The content directly depends on the bit fields EGPR1_SEL, GPR1_SEL of register GTM0TIMixCTRL.
3. The content of this register can only be written in TIM channel mode TGPS.

### 25.10.7.7 GTM0TIMixCNT (i = 0,1, x = 0 to 7)

**Access:** This register can be read in 32-bit units.

**Address:**

| | |
|---|---|
| GTM0TIM00CNT: <GTM_base> + $01008_H$, | GTM0TIM10CNT: <GTM_base> + $01808_H$ |
| GTM0TIM01CNT: <GTM_base> + $01088_H$, | GTM0TIM11CNT: <GTM_base> + $01888_H$ |
| GTM0TIM02CNT: <GTM_base> + $01108_H$, | GTM0TIM12CNT: <GTM_base> + $01908_H$ |
| GTM0TIM03CNT: <GTM_base> + $01188_H$, | GTM0TIM13CNT: <GTM_base> + $01988_H$ |
| GTM0TIM04CNT: <GTM_base> + $01208_H$, | GTM0TIM14CNT: <GTM_base> + $01A08_H$ |
| GTM0TIM05CNT: <GTM_base> + $01288_H$, | GTM0TIM15CNT: <GTM_base> + $01A88_H$ |
| GTM0TIM06CNT: <GTM_base> + $01308_H$, | GTM0TIM16CNT: <GTM_base> + $01B08_H$ |
| GTM0TIM07CNT: <GTM_base> + $01388_H$, | GTM0TIM17CNT: <GTM_base> + $01B88_H$ |

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CNT | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNT | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.77 GTM0TIMixCNT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. |
| 23 to 0 | CNT | Actual SMU counter value<br>**NOTE**<br>The meaning of this value depends on the configured mode:<br>TPWM = actual duration of PWM signal.<br>TPIM = actual duration of all pulses (sum of pulses).<br>TIEM = actual number of received edges.<br>TIPM = actual number of received edges. |

### 25.10.7.8 GTM0TIMixCNTS (i = 0,1, x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:**

| | |
|---|---|
| GTM0TIM00CNTS: <GTM_base> + $01010_H$, | GTM0TIM10CNTS: <GTM_base> + $01810_H$ |
| GTM0TIM01CNTS: <GTM_base> + $01090_H$, | GTM0TIM11CNTS: <GTM_base> + $01890_H$ |
| GTM0TIM02CNTS: <GTM_base> + $01110_H$, | GTM0TIM12CNTS: <GTM_base> + $01910_H$ |
| GTM0TIM03CNTS: <GTM_base> + $01190_H$, | GTM0TIM13CNTS: <GTM_base> + $01990_H$ |
| GTM0TIM04CNTS: <GTM_base> + $01210_H$, | GTM0TIM14CNTS: <GTM_base> + $01A10_H$ |
| GTM0TIM05CNTS: <GTM_base> + $01290_H$, | GTM0TIM15CNTS: <GTM_base> + $01A90_H$ |
| GTM0TIM06CNTS: <GTM_base> + $01310_H$, | GTM0TIM16CNTS: <GTM_base> + $01B10_H$ |
| GTM0TIM07CNTS: <GTM_base> + $01390_H$, | GTM0TIM17CNTS: <GTM_base> + $01B90_H$ |

**Value after reset:** $0X00\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECNT | | | | | | | | CNTS | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNTS | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.78 GTM0TIMixCNTS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | ECNT | Edge counter.<br>**NOTE**<br>The ECNT counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of ECNT. |
| 23 to 0 | CNTS | Counter shadow register. |

**NOTES**

1. The ECNT register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.
2. The content of this register has different meaning for the TIM channels modes. The content depends directly on the bit field CNTS_SEL of register GTM0TIMixCTRL.
3. The register GTM0TIMixCNTS is only writable in TIPM,TBCM and TGPS mode.

### 25.10.7.9 GTM0TIMixIRQNOTIFY (i = 0,1, x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0TIM00IRQNOTIFY: <GTM_base> + $0102C_H$, GTM0TIM10IRQNOTIFY: <GTM_base> + $0182C_H$
GTM0TIM01IRQNOTIFY: <GTM_base> + $010AC_H$, GTM0TIM11IRQNOTIFY: <GTM_base> + $018AC_H$
GTM0TIM02IRQNOTIFY: <GTM_base> + $0112C_H$, GTM0TIM12IRQNOTIFY: <GTM_base> + $0192C_H$
GTM0TIM03IRQNOTIFY: <GTM_base> + $011AC_H$, GTM0TIM13IRQNOTIFY: <GTM_base> + $019AC_H$
GTM0TIM04IRQNOTIFY: <GTM_base> + $0122C_H$, GTM0TIM14IRQNOTIFY: <GTM_base> + $01A2C_H$
GTM0TIM05IRQNOTIFY: <GTM_base> + $012AC_H$, GTM0TIM15IRQNOTIFY: <GTM_base> + $01AAC_H$
GTM0TIM06IRQNOTIFY: <GTM_base> + $0132C_H$, GTM0TIM16IRQNOTIFY: <GTM_base> + $01B2C_H$
GTM0TIM07IRQNOTIFY: <GTM_base> + $013AC_H$, GTM0TIM17IRQNOTIFY: <GTM_base> + $01BAC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | GLITCHDET | TODET | GPROFL | CNTOFL | ECNTOFL | NEWVAL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.79 GTM0TIMixIRQNOTIFY Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 | GLITCHDET | Glitch detected on channel x, (x = 0 to 7).<br>0: No glitch detected for last edge<br>1: Glitch detected for last edge<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 4 | TODET | Timeout reached for input signal of channel x, (x = 0 to 7). See bit 0. |
| 3 | GPROFL | GPR0 and GPR1 data overflow, old data not read out before new data has arrived at input pin, (x = 0 to 7). See bit 0. |
| 2 | CNTOFL | SMU CNT counter overflow of channel x, (x = 0 to 7). See bit 0. |
| 1 | ECNTOFL | ECNT counter overflow of channel x, (x = 0 to 7). See bit 0. |
| 0 | NEWVAL | New measurement value detected by in channel x (x =0 to 7)<br>0: No event was occurred<br>1: NEWVAL was occurred on the TIM channel<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

## 25.10.7.10GTM0TIMixIRQEN (i = 0,1, x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:**

| | |
|---|---|
| GTM0TIM00IRQEN: <GTM_base> + $01030_H$, | GTM0TIM10IRQEN: <GTM_base> + $01830_H$ |
| GTM0TIM01IRQEN: <GTM_base> + $010B0_H$, | GTM0TIM11IRQEN: <GTM_base> + $018B0_H$ |
| GTM0TIM02IRQEN: <GTM_base> + $01130_H$, | GTM0TIM12IRQEN: <GTM_base> + $01930_H$ |
| GTM0TIM03IRQEN: <GTM_base> + $011B0_H$, | GTM0TIM13IRQEN: <GTM_base> + $019B0_H$ |
| GTM0TIM04IRQEN: <GTM_base> + $01230_H$, | GTM0TIM14IRQEN: <GTM_base> + $01A30_H$ |
| GTM0TIM05IRQEN: <GTM_base> + $012B0_H$, | GTM0TIM15IRQEN: <GTM_base> + $01AB0_H$ |
| GTM0TIM06IRQEN: <GTM_base> + $01330_H$, | GTM0TIM16IRQEN: <GTM_base> + $01B30_H$ |
| GTM0TIM07IRQEN: <GTM_base> + $013B0_H$, | GTM0TIM17IRQEN: <GTM_base> + $01BB0_H$ |

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | GLITCHDET_IRQ_EN | TODET_IRQ_EN | GPROFL_IRQ_EN | CNTOFL_IRQ_EN | ECNTOFL_IRQ_EN | NEWVAL_IRQ_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.80 GTM0TIMixIRQEN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 | GLITCHDET_ IRQ_ EN | TIM_GLITCHDETx_IRQ interrupt enable, see bit 0. |
| 4 | TODET_IRQ_ EN | TIM_TODETx_IRQ interrupt enable, see bit 0. |
| 3 | GPROFL_IRQ_ EN | TIM_GPROFL_IRQ interrupt enable, see bit 0. |
| 2 | CNTOFL_IRQ_ EN | TIM_CNTOFLx_IRQ interrupt enable, see bit 0. |
| 1 | ECNTOFL_IRQ _EN | TIM_ECNTOFLx_IRQ interrupt enable, see bit 0. |
| 0 | NEWVAL_IRQ_ EN | TIM_NEWVALx_IRQ interrupt enable<br>0: Disable interrupt, interrupt is not visible outside GTM-IP<br>1: Enable interrupt, interrupt is visible outside GTM-IP |

## 25.10.7.11GTM0TIMixIRQFORCINT (i = 0,1, x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0TIM00IRQFORCINT: <GTM_base> + $01034_H$, GTM0TIM10IRQFORCINT: <GTM_base> + $01834_H$
GTM0TIM01IRQFORCINT: <GTM_base> + $010B4_H$, GTM0TIM11IRQFORCINT: <GTM_base> + $018B4_H$
GTM0TIM02IRQFORCINT: <GTM_base> + $01134_H$, GTM0TIM12IRQFORCINT: <GTM_base> + $01934_H$
GTM0TIM03IRQFORCINT: <GTM_base> + $011B4_H$, GTM0TIM13IRQFORCINT: <GTM_base> + $019B4_H$
GTM0TIM04IRQFORCINT: <GTM_base> + $01234_H$, GTM0TIM14IRQFORCINT: <GTM_base> + $01A34_H$
GTM0TIM05IRQFORCINT: <GTM_base> + $012B4_H$, GTM0TIM15IRQFORCINT: <GTM_base> + $01AB4_H$
GTM0TIM06IRQFORCINT: <GTM_base> + $01334_H$, GTM0TIM16IRQFORCINT: <GTM_base> + $01B34_H$
GTM0TIM07IRQFORCINT: <GTM_base> + $013B4_H$, GTM0TIM17IRQFORCINT: <GTM_base> + $01BB4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | TRG_GLITCHDET | TRG_TODET | TRG_GPROFL | TRG_CNTOFL | TRG_ECNTOFL | TRG_NEWVAL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.81 GTM0TIMixIRQFORCINT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 | TRG_GLITCHDET | Trigger GLITCHDET bit in GTM0TIMixIRQNOTIFY register by software, see bit 0. |
| 4 | TRG_TODET | Trigger TODET bit in GTM0TIMixIRQNOTIFY register by software, see bit 0. |
| 3 | TRG_GPROFL | Trigger GPROFL bit in GTM0TIMixIRQNOTIFY register by software, see bit 0. |
| 2 | TRG_CNTOFL | Trigger CNTOFL bit in GTM0TIMixIRQNOTIFY register by software, see bit 0. |
| 1 | TRG_ECNTOFL | Trigger ECNTOFL bit in GTM0TIMixIRQNOTIFY register by software, see bit 0. |
| 0 | TRG_NEWVAL | Trigger NEWVAL bit in GTM0TIMixIRQNOTIFY register by software<br>0: No interrupt triggering<br>1: Assert corresponding field in GTM0TIMixIRQNOTIFY register<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of register GTM0GTMCTRL |

## 25.10.7.12GTM0TIMixIRQMODE (i = 0,1, x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0TIM00IRQMODE: <GTM_base> + $01038_H$, GTM0TIM10IRQMODE: <GTM_base> + $01838_H$
GTM0TIM01IRQMODE: <GTM_base> + $010B8_H$, GTM0TIM11IRQMODE: <GTM_base> + $018B8_H$
GTM0TIM02IRQMODE: <GTM_base> + $01138_H$, GTM0TIM12IRQMODE: <GTM_base> + $01938_H$
GTM0TIM03IRQMODE: <GTM_base> + $011B8_H$, GTM0TIM13IRQMODE: <GTM_base> + $019B8_H$
GTM0TIM04IRQMODE: <GTM_base> + $01238_H$, GTM0TIM14IRQMODE: <GTM_base> + $01A38_H$
GTM0TIM05IRQMODE: <GTM_base> + $012B8_H$, GTM0TIM15IRQMODE: <GTM_base> + $01AB8_H$
GTM0TIM06IRQMODE: <GTM_base> + $01338_H$, GTM0TIM16IRQMODE: <GTM_base> + $01B38_H$
GTM0TIM07IRQMODE: <GTM_base> + $013B8_H$, GTM0TIM17IRQMODE: <GTM_base> + $01BB8_H$

**Value after reset:** $0000\ 000X_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IRQ_MODE | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 25.82 GTM0TIMixIRQMODE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1, 0 | IRQ_MODE | IRQ mode selection<br>00: Level mode<br>01: Pulse mode<br>10: Pulse-Notify mode<br>11: Single-Pulse mode<br>**NOTE**<br>The interrupt modes are described in **Section 25.6.5, GTM-IP Interrupt Concept** . |

## 25.10.7.13GTM0TIMiRST (i = 0,1)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0TIM0RST: <GTM_base> + $0107C_H$
GTM0TIM1RST: <GTM_base> + $0187C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RST_CH7 | RST_CH6 | RST_CH5 | RST_CH4 | RST_CH3 | RST_CH2 | RST_CH1 | RST_CH0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.83 GTM0TIMiRST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 7 | RST_CH7 | Software reset of channel 7, see bit 0. |
| 6 | RST_CH6 | Software reset of channel 6, see bit 0. |
| 5 | RST_CH5 | Software reset of channel 5, see bit 0. |
| 4 | RST_CH4 | Software reset of channel 4, see bit 0. |
| 3 | RST_CH3 | Software reset of channel 3, see bit 0. |
| 2 | RST_CH2 | Software reset of channel 2, see bit 0. |
| 1 | RST_CH1 | Software reset of channel 1, see bit 0. |
| 0 | RST_CH0 | Software reset of channel 0<br>0: No action<br>1: Reset channel 0<br>**NOTE**<br>This bit is cleared automatically after write by CPU. The channel registers are set to their reset values and channel operation is stopped immediately. |

**NOTE**

Please note, that the RST field width of this register depends on the number of implemented channels m within this submodule. This register description represents a register layout for m = 8.

## 25.10.7.14GTM0TIMiINSRC (i = 0,1)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0TIM0INSRC: <GTM_base> + $01078_H$
GTM0TIM1INSRC: <GTM_base> + $01878_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MODE_7 | | VAL_7 | | MODE_6 | | VAL_6 | | MODE_5 | | VAL_5 | | MODE_4 | | VAL_4 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MODE_3 | | VAL_3 | | MODE_2 | | VAL_2 | | MODE_1 | | VAL_1 | | MODE_0 | | VAL_0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.84 GTM0TIMiINSRC Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | MODE_7 | Input source to Channel 7, see bits 3, 2. |
| 29, 28 | VAL_7 | Value to be fed to Channel 7, see bits 1, 0. |
| 27, 26 | MODE_6 | Input source to Channel 6, see bits 3, 2. |
| 25, 24 | VAL_6 | Value to be fed to Channel 6, see bits 1, 0. |
| 23, 22 | MODE_5 | Input source to Channel 5, see bits 3, 2. |
| 21, 20 | VAL_5 | Value to be fed to Channel 5, see bits 1, 0. |
| 19, 18 | MODE_4 | Input source to Channel 4, see bits 3, 2. |
| 17, 16 | VAL_4 | Value to be fed to Channel 4, see bits 1, 0. |
| 15, 14 | MODE_3 | Input source to Channel 3, see bits 3, 2. |
| 13, 12 | VAL_3 | Value to be fed to Channel 3, see bits 1, 0. |
| 11, 10 | MODE_2 | Input source to Channel 2, see bits 3, 2. |
| 9, 8 | VAL_2 | Value to be fed to Channel 2, see bits 1, 0. |
| 7, 6 | MODE_1 | Input source to Channel 1, see bits 3, 2. |
| 5, 4 | VAL_1 | Value to be fed to Channel 1, see bits 1, 0. |
| 3, 2 | MODE_0 | Input source to Channel 0<br>multicore encoding in use (MODE_x(1) defines the state of the signal)<br>00: State is 0 (ignore write access)<br>01: Change state to 0<br>10: Change state to 1<br>11: State is 1 (ignore write access)<br>Function table:<br>MODE_x(1) = 0 , VAL_x(1) = 0: The input signal defined by bit field CICTRL of the TIM channel is used as input source.<br>MODE_x(1) = 0 , VAL_x(1) = 1: The signal TIM_AUX_IN of the TIM channel is used as input source.<br>MODE_x(1) = 1 : The state VAL_x(1) defines the input level for the TIM channel.<br>**NOTE**<br>Any read access to a MODE_x bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored. |

**Table 25.84 GTM0TIMiINSRC Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1, 0 | VAL_0 | Value to be fed to Channel 0<br>multicore encoding in use (VAL_x(1) defines the state of the signal)<br>00: State is 0 (ignore write access)<br>01: Change state to 0<br>10: Change state to 1<br>11: State is 1 (ignore write access)<br>Function depends on the combination of VAL_x(1) and MODE_x(1) see MODE_0 description.<br>**NOTE**<br>Any read access to a VAL_x bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored. |

## 25.10.7.15GTM0TIMixEIRQEN (i = 0,1, x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0TIM00EIRQEN: <GTM_base> + $0103C_H$, GTM0TIM10EIRQEN: <GTM_base> + $0183C_H$
GTM0TIM01EIRQEN: <GTM_base> + $010BC_H$, GTM0TIM11EIRQEN: <GTM_base> + $018BC_H$
GTM0TIM02EIRQEN: <GTM_base> + $0113C_H$, GTM0TIM12EIRQEN: <GTM_base> + $0193C_H$
GTM0TIM03EIRQEN: <GTM_base> + $011BC_H$, GTM0TIM13EIRQEN: <GTM_base> + $019BC_H$
GTM0TIM04EIRQEN: <GTM_base> + $0123C_H$, GTM0TIM14EIRQEN: <GTM_base> + $01A3C_H$
GTM0TIM05EIRQEN: <GTM_base> + $012BC_H$, GTM0TIM15EIRQEN: <GTM_base> + $01ABC_H$
GTM0TIM06EIRQEN: <GTM_base> + $0133C_H$, GTM0TIM16EIRQEN: <GTM_base> + $01B3C_H$
GTM0TIM07EIRQEN: <GTM_base> + $013BC_H$, GTM0TIM17EIRQEN: <GTM_base> + $01BBC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | GLITCHDET_EIRQ_EN | TODET_EIRQ_EN | GPROFL_EIRQ_EN | CNTOFL_EIRQ_EN | ECNTOFL_EIRQ_EN | NEWVAL_EIRQ_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.85 GTM0TIMixEIRQEN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 | GLITCHDET_EIRQ_EN | TIM_GLITCHDETx_IRQ interrupt enable, see bit 0. |
| 4 | TODET_EIRQ_EN | TIM_TODETx_IRQ interrupt enable, see bit 0. |
| 3 | GPROFL_EIRQ_EN | TIM_GPROFL_IRQ interrupt enable, see bit 0. |
| 2 | CNTOFL_EIRQ_EN | TIM_CNTOFLx_IRQ interrupt enable, see bit 0. |
| 1 | ECNTOFL_EIRQ_EN | TIM_ECNTOFLx_IRQ interrupt enable, see bit 0. |
| 0 | NEWVAL_EIRQ_EN | TIM_NEWVALx_EIRQ error interrupt enable<br>0: Disable error interrupt, error interrupt is not visible outside GTM-IP<br>1: Enable error interrupt, error interrupt is visible outside GTM-IP |

## 25.10.7.16GTM0TIMixTDUV (i = 0,1, x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:**

| | |
|---|---|
| GTM0TIM00TDUV: <GTM_base> + $01018_H$, | GTM0TIM10TDUV: <GTM_base> + $01818_H$ |
| GTM0TIM01TDUV: <GTM_base> + $01098_H$, | GTM0TIM11TDUV: <GTM_base> + $01898_H$ |
| GTM0TIM02TDUV: <GTM_base> + $01118_H$, | GTM0TIM12TDUV: <GTM_base> + $01918_H$ |
| GTM0TIM03TDUV: <GTM_base> + $01198_H$, | GTM0TIM13TDUV: <GTM_base> + $01998_H$ |
| GTM0TIM04TDUV: <GTM_base> + $01218_H$, | GTM0TIM14TDUV: <GTM_base> + $01A18_H$ |
| GTM0TIM05TDUV: <GTM_base> + $01298_H$, | GTM0TIM15TDUV: <GTM_base> + $01A98_H$ |
| GTM0TIM06TDUV: <GTM_base> + $01318_H$, | GTM0TIM16TDUV: <GTM_base> + $01B18_H$ |
| GTM0TIM07TDUV: <GTM_base> + $01398_H$, | GTM0TIM17TDUV: <GTM_base> + $01B98_H$ |

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TCS | | | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TOV | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.86 GTM0TIMixTDUV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 30 to 28 | TCS | Timeout Clock selection<br>000: CMU_CLK0 selected<br>001: CMU_CLK1 selected<br>010: CMU_CLK2 selected<br>011: CMU_CLK3 selected<br>100: CMU_CLK4 selected<br>101: CMU_CLK5 selected<br>110: CMU_CLK6 selected<br>111: CMU_CLK7 selected |
| 27 to 8 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 7 to 0 | TOV | Time out duration for channel x (x = 0 to 7). |

## 25.10.7.17GTM0TIMixTDUC (i = 0,1, x = 0 to 7)

**Access:** This register can be read in 32-bit units.

**Address:**

| | |
|---|---|
| GTM0TIM00TDUC: <GTM_base> + $01014_H$, | GTM0TIM10TDUC: <GTM_base> + $01814_H$ |
| GTM0TIM01TDUC: <GTM_base> + $01094_H$, | GTM0TIM11TDUC: <GTM_base> + $01894_H$ |
| GTM0TIM02TDUC: <GTM_base> + $01114_H$, | GTM0TIM12TDUC: <GTM_base> + $01914_H$ |
| GTM0TIM03TDUC: <GTM_base> + $01194_H$, | GTM0TIM13TDUC: <GTM_base> + $01994_H$ |
| GTM0TIM04TDUC: <GTM_base> + $01214_H$, | GTM0TIM14TDUC: <GTM_base> + $01A14_H$ |
| GTM0TIM05TDUC: <GTM_base> + $01294_H$, | GTM0TIM15TDUC: <GTM_base> + $01A94_H$ |
| GTM0TIM06TDUC: <GTM_base> + $01314_H$, | GTM0TIM16TDUC: <GTM_base> + $01B14_H$ |
| GTM0TIM07TDUC: <GTM_base> + $01394_H$, | GTM0TIM17TDUC: <GTM_base> + $01B94_H$ |

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TO_CNT | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.87 GTM0TIMixTDUC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | These bits are always read as 0. |
| 7 to 0 | TO_CNT | Current Timeout value for channel x (x = 0 to 7). |

## 25.10.7.18GTM0TIMixECNT (i = 0,1, x = 0 to 7)

**Access:** This register can be read in 32-bit units.

**Address:** GTM0TIM00ECNT: <GTM_base> + $0100C_H$, GTM0TIM10ECNT: <GTM_base> + $0180C_H$
GTM0TIM01ECNT: <GTM_base> + $0108C_H$, GTM0TIM11ECNT: <GTM_base> + $0188C_H$
GTM0TIM02ECNT: <GTM_base> + $0110C_H$, GTM0TIM12ECNT: <GTM_base> + $0190C_H$
GTM0TIM03ECNT: <GTM_base> + $0118C_H$, GTM0TIM13ECNT: <GTM_base> + $0198C_H$
GTM0TIM04ECNT: <GTM_base> + $0120C_H$, GTM0TIM14ECNT: <GTM_base> + $01A0C_H$
GTM0TIM05ECNT: <GTM_base> + $0128C_H$, GTM0TIM15ECNT: <GTM_base> + $01A8C_H$
GTM0TIM06ECNT: <GTM_base> + $0130C_H$, GTM0TIM16ECNT: <GTM_base> + $01B0C_H$
GTM0TIM07ECNT: <GTM_base> + $0138C_H$, GTM0TIM17ECNT: <GTM_base> + $01B8C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECNT | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.88 GTM0TIMixECNT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. |
| 15 to 0 | ECNT | Edge counter<br>**NOTE**<br>If TIM channel is disabled the content of ECNT gets frozen. A read will auto clear the bits [15:1]. Further read accesses to ECNT will show on Bit 0 the actual input signal value of the channel. |

### 25.10.7.19 GTM0TIM00ECTRL (i = 0,1, x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0TIM00ECTRL: <GTM_base> + $01028_H$,
GTM0TIM01ECTRL: <GTM_base> + $010A8_H$,
GTM0TIM02ECTRL: <GTM_base> + $01128_H$,
GTM0TIM03ECTRL: <GTM_base> + $011A8_H$,
GTM0TIM04ECTRL: <GTM_base> + $01228_H$,
GTM0TIM05ECTRL: <GTM_base> + $012A8_H$,
GTM0TIM06ECTRL: <GTM_base> + $01328_H$,
GTM0TIM07ECTRL: <GTM_base> + $013A8_H$,
GTM0TIM10ECTRL: <GTM_base> + $01828_H$
GTM0TIM11ECTRL: <GTM_base> + $018A8_H$
GTM0TIM12ECTRL: <GTM_base> + $01928_H$
GTM0TIM13ECTRL: <GTM_base> + $019A8_H$
GTM0TIM14ECTRL: <GTM_base> + $01A28_H$
GTM0TIM15ECTRL: <GTM_base> + $01AA8_H$
GTM0TIM16ECTRL: <GTM_base> + $01B28_H$
GTM0TIM17ECTRL: <GTM_base> + $01BA8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | EXT_CAP_SRC | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 25.89 GTM0TIM00ECTRL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 3 to 0 | EXT_CAP_SRC | Defines selected source for triggering the EXT_CAPTURE functionality.<br>0000: NEW_VAL_IRQ of following channel selected<br>0001: AUX_IN selected<br>0010: CNTOFL_IRQ of following channel selected<br>0011 and CICTRL = 1: Use signal TIM_IN(x) as input for channel x<br>0011 and CICTRL = 0: Use signal TIM_IN(x-1) as input for channel x (or TIM_IN(m-1) if x is 0)<br>0100: ECNTOFL_IRQ of following channel selected<br>0101:TODET_IRQ of following channel selected<br>0110: GLITCHDET_IRQ of following channel selected<br>0111: GPROFL_IRQ of following channel selected<br>1000: cmu_clk selected by CLK_SEL of following channel<br>1001: REDGE_DET of following channel selected<br>1010: FEDGE_DET of following channel selected<br>1011: Logical or of (FEDGE_DET, REDGE_DET) of following channel selected<br>**NOTE**<br>Undefined values will not be written and AEI_STATUS will signal “10” |

## 25.10.7.20GTM0TIMiINPVAL (i = 0,1)

**Access:** This register can be read in 32-bit units.

**Address:** GTM0TIM0INPVAL: <GTM_base> + $01074_H$
GTM0TIM1INPVAL: <GTM_base> + $01874_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TIM_IN | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F_IN | | | | | | | | F_OUT | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.90 GTM0TIMiINPVAL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. |
| 23 to 16 | TIM_IN | Signals after TIM input signal synchronization |
| 15 to 8 | F_IN | Signals after INPSRC selection, before TIM FLT unit |
| 7 to 0 | F_OUT | Signals after TIM FLT unit |

# 25.11 ARU-connected Timer Output Module (ATOM)

## 25.11.1 Overview

The ARU-connected Timer Output Module (ATOM) is able to generate complex output signals without CPU interaction due to its connectivity to the ARU. Typically, output signal characteristics are provided over the ARU connection through submodules connected to ARU like e.g. the MCS, DPLL or PSM. Each ATOM submodule contains eight output channels which can operate independently from each other in several configurable operation modes. A block diagram of the ATOM submodule is depicted in **Figure 25.32**.

The following design variables are used inside this chapter. See **Section 25.18, GTM Device 207** and **Section 25.19, GTM Device 208** for correct value.

**NOTE**

cCATO : ATOM channel count; number of channels per instance – 1

### 25.11.1.1 ATOM block diagram

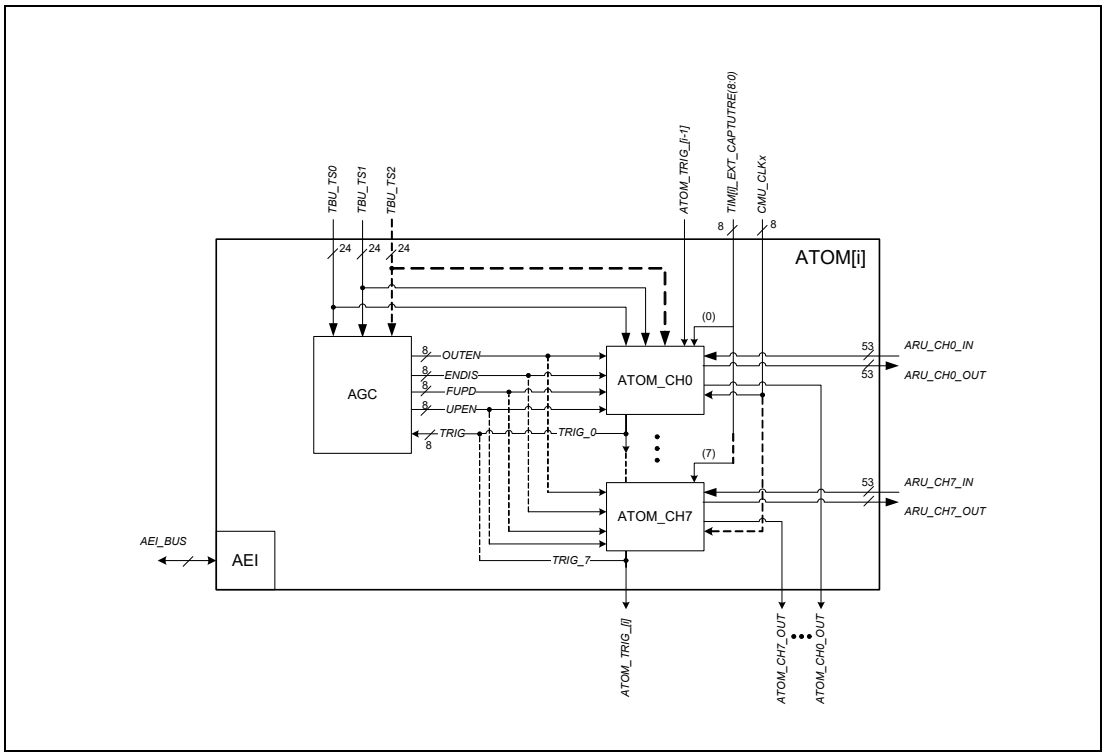



**Figure 25.32 ATOM block diagram**

The architecture of the ATOM submodule is similar to the TOM submodule, but there are some differences. First, the ATOM integrates only eight output channels. Hence, there exists one ATOM Global Control subunit (AGC) for the ATOM channels. The ATOM is connected to the ARU and can set up individual read requests from the ARU and write requests to the ARU. Furthermore, the ATOM channels are able to generate signals on behalf of time stamps and the ATOM channels are able to generate a serial output signal on behalf of an internal shift register.

Each ATOM channel provides five modes of operation:

- ATOM Signal Output Mode Immediate (SOMI)
- ATOM Signal Output Mode Compare (SOMC)
- ATOM Signal Output Mode PWM (SOMP)
- ATOM Signal Output Mode Serial (SOMS)
- ATOM Signal Output Mode Buffered Compare (SOMB)

These modes are described in more detail in **Section 25.11.3, ATOM Channel modes** .

The ATOM channels' operation registers (e.g. counter, compare registers) are 24 bit wide. Moreover, the input clocks for the ATOM channels come from the configurable CMU_CLKx signals of the CMU submodule. This gives the freedom to select a programmable input clock for the ATOM channel counters. The ATOM channel is able to generate a serial bit stream, which is shifted out at the ATOM[i]_CH[x]_OUT output. When configured in this serial shift mode (SOMS) the selected CMU clock defines the shift frequency.

Each ATOM channel provides a so called operation and shadow register set. With this architecture it is possible to work with the operation register set, while the shadow register set can be reloaded with new parameters over CPU and/or ARU.

When update via ARU is selected, it is possible to configure if both shadow registers are updated via ARU or only one of the shadow registers is updated for SOMP mode.

On the other hand, the shadow registers can be used to provide data to the ARU when one or both of the compare units inside an ATOM channel match. This feature is only applicable in SOMC mode.

In TOM channels it is possible to reload the content of the operation registers with the content of the corresponding shadow registers and change the clock input signal for the counter register simultaneously. This simultaneous change of the input clock frequency together with reloading the operation registers is also implemented in the ATOM channels.

In addition to the feature that the CPU can select another CMU_CLKx during operation (i.e. updating the shadow register bit field CLK_SRC_SR of the GTM0ATOMixCTRL register), the selection can also be changed via the ARU. Then, for the clock source update, the ACBI register bits of the GTM0ATOMixSTAT register are used as a shadow register for the new clock source.

In general, the behavior of the compare units CCU0 and CCU1 and the output signal behavior is controlled with the ACB bit field inside the GTM0ATOMixCTRL register when the ARU connection is disabled and the behavior is controlled via ARU through the ACBI bit field of the GTM0ATOMixSTAT register, when the ARU is enabled.

Since the ATOM is connected to the ARU, the shadow registers of an ATOM channel can be reloaded via the ARU connection or via CPU over its AEI interface. When loaded via the ARU interface, the shadow registers act as a buffer between the ARU and the channel operation registers. Thus, a new parameter set for a PWM can be reloaded via ARU into the shadow registers, while the operation registers work on the actual parameter set.

The trigger signal ATOM_TRIG_[i–1] of ATOM instance i comes from the preceding instance i–1, the trigger ATOM_TRIG_[i] is routed to succeeding instance i+1.

Note, ATOM0 is connected to its own output ATOM_TRIG_0, i.e. the last channel of ATOM instance 0 can trigger the first channel of ATOM instance 0 (this path is registered, which means delayed by one SYS_CLK period).

### 25.11.1.2 ATOM Global control (AGC)

Synchronous start and stop of more than one output channel is possible with the AGC subunit. This subunit has the same functionality as the TGC subunit of the TOM submodule. For a description of the AGC subunit functionality, please refer therefore to chapter **(2) AGC Subunit**.

#### (1) Overview

There exist one global channel control unit (AGC) to drive a number of individual ATOM channels synchronously by external or internal events.

An AGC can drive up to eight ATOM channels.

The ATOM submodule supports four different kinds of signalling mechanisms:

- Global enable/disable mechanism for each ATOM channel with control register GTM0ATOMiAGCENDISCTRL and status register GTM0ATOMiAGCENDISSTAT
- Global output enable mechanism for each ATOM channel with control register GTM0ATOMiAGCOUTENCTRL and status register GTM0ATOMiAGCOUTENSTAT
- Global force update mechanism for each ATOM channel with control register GTM0ATOMiAGCFUPDCTRL
- Update enable of the register CM0, CM1 and CLK_SRC for each ATOM channel with the control bit field UPEN_CTRL[z] of GTM0ATOMiAGCGLBCTRL

#### (2) AGC Subunit

Each of the first three individual mechanisms (enable/disable of the channel, output enable and force update) can be driven by three different trigger sources.

The three trigger sources are :

- the host CPU (bit HOST_TRIG of register GTM0ATOMiAGCGLBCTRL)
- the TBU time stamp (signal TBU_TS0..2 if available)
- the internal trigger signal TRIG (bunch of trigger signals TRIG_[x]) which can be either the trigger TRIG_CCU0 of channel x, the trigger of preceeding channel x-1 (i.e signal TRIG_[x–1]) or the external trigger TIM_EXT_CAPTURE(x) of assigned TIM channel x.

The first way is to trigger the control mechanism by a direct register write access via host CPU (bit HOST_TRIG of register AOM[i]_AGC_GLB_CTRL).

The second way is provided by a compare match trigger on behalf of a specified time base coming from the module TBU (selected by bits TBU_SEL) and the time stamp compare value defined in the bit field ACT_TB of register GTM0ATOMiAGCACTTB.

Note, a signed compare of ACT_TB and selected TBU_TS[x] is performed.

The third possibility is the input TRIG (bunch of trigger signals TRIG_[x]) coming from the ATOM channels 0 to 7.

The corresponding trigger signal TRIG_[x] coming from channel [x] can be masked by the register GTM0ATOMiAGCINTTRIG.

To enable or disable each individual ATOM channel, the registers GTM0ATOMiAGCENDISCTRL and/or GTM0ATOMiAGCENDISSTAT have to be used.

The register GTM0ATOMiAGCENDISSTAT controls directly the signal ENDIS. A write access to this register is possible.

The register GTM0ATOMiAGCENDISCTRL is a shadow register that overwrites the value of register AOM[i]_AGC_ENDIS_STAT if one of the three trigger conditions matches.

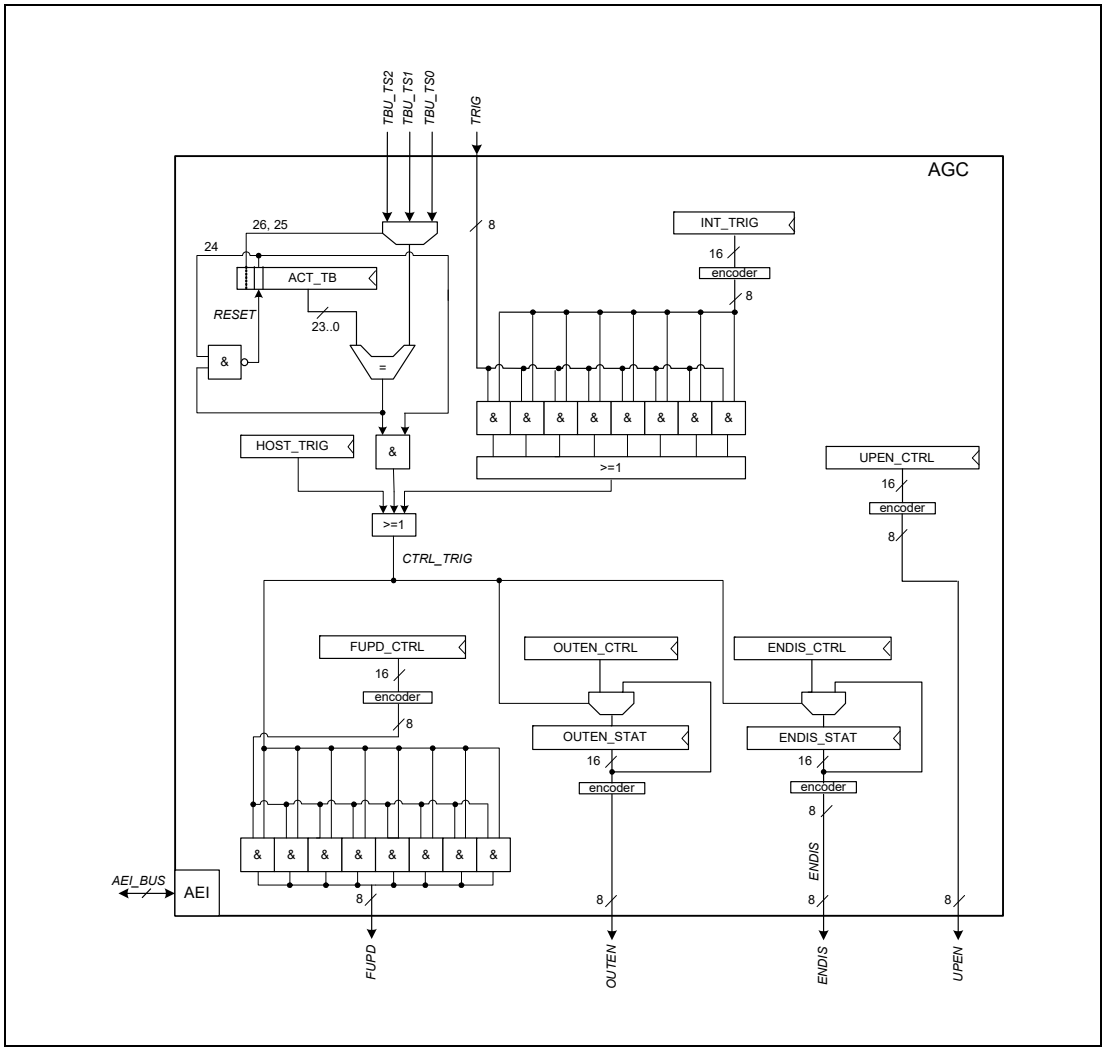



**Figure 25.33 ATOM Global channel control mechanism**

The output of the individual ATOM channels can be controlled using the register GTM0ATOMiAGCOUTENCTRL and GTM0ATOMiAGCOUTENSTAT.

The register GTM0ATOMiAGCOUTENSTAT controls directly the signal OUTEN. A write access to this register is possible.

The register GTM0ATOMiAGCOUTENCTRL is a shadow register that overwrites the value of register GTM0ATOMiAGCOUTENSTAT if one of the three trigger conditions matches.

If a ATOM channel is disabled by the register GTM0ATOMiAGCOUTENSTAT, the actual value of the channel output at ATOM_CH[x]_OUT is defined by the signal level bit (SL) defined in the channel control register GTM0ATOMixCTRL.

If the output is enabled, the output at ATOM_CH[x]_OUT depends on value of Flip-Flop SOUR.

The register GTM0ATOMiAGCFUPDCTRL defines which of the ATOM channels receive a FORCE_UPDATE event if the trigger signal CTRL_TRIG is raised.

The register bits UPEN_CTRL[x] defines for which ATOM channel the update of the working register CM0, CM1 and CLK_SRC by the corresponding shadow register SR0, SR1 and CLK_SRC_SR is

enabled. If update is enabled, the register CM0, CM1 and CLK_SRC will be updated on reset of counter register CN0 (see **Figure 25.34**).

### (3) ATOM Channel mode overview

Each ATOM channel offers the following different operation modes:

In ATOM Signal Output Mode Immediate (SOMI), the ATOM channels generate an output signal immediately after receiving an ARU word according to the two signal level output bits of the ARU word received through the ACBI bit field. Due to the fact, that the ARU destination channels are served in a round robin order, the output signal can jitter in this mode with a jitter of the ARU round trip time.

In ATOM Signal Output Mode Compare (SOMC), the ATOM channel generates an output signal on behalf of time stamps that are located in the ATOM operation registers. These time stamps are compared with the time stamps, the TBU generates. The ATOM is able to receive new time stamps either by CPU or via the ARU. The new time stamps are directly loaded into the channels operation register. The shadow registers are used as capture registers for two time base values, when a compare match of the channels operation registers occurs.

In ATOM Signal Output Mode PWM (SOMP), the ATOM channel is able to generate simple and complex PWM output signals like the TOM submodule by comparing its operation registers with a submodule internal counter. In difference to the TOM, the ATOM shadow registers can be reloaded by the CPU and by the ARU in the background, while the channel operates on the operation registers.

In ATOM Signal Output Mode Serial (SOMS), the ATOM channel generates a serial output bit stream on behalf of a shift register. The number of bits shifted and the shift direction is configurable. The shift frequency is determined by one of the CMU_CLKx clock signals. See **Section 25.11.3.4, ATOM Signal Output Mode Serial (SOMS)** for further details.

In ATOM Signal Output Buffered Compare (SOMB), the ATOM channel generates an output signal on behalf of time stamps that located in the ATOM operation registers. These time stamps are compared with the time stamps, the TBU generates. The ATOM is able to receive new compare values either by CPU or via the ARU. The new compare values received via ARU are stored first in the shadow register and only if previous compare match is occurred, the operation register are updated with the content of the shadow register.

## 25.11.2 ATOM Channel architecture

Each ATOM channel is able to generate output signals according to four operation modes. The architecture of the ATOM channels is similar to the architecture of the TOM channels. The general architecture of an ATOM channel is depicted in **Figure 25.34**.

### 25.11.2.1 ATOM channel architecture

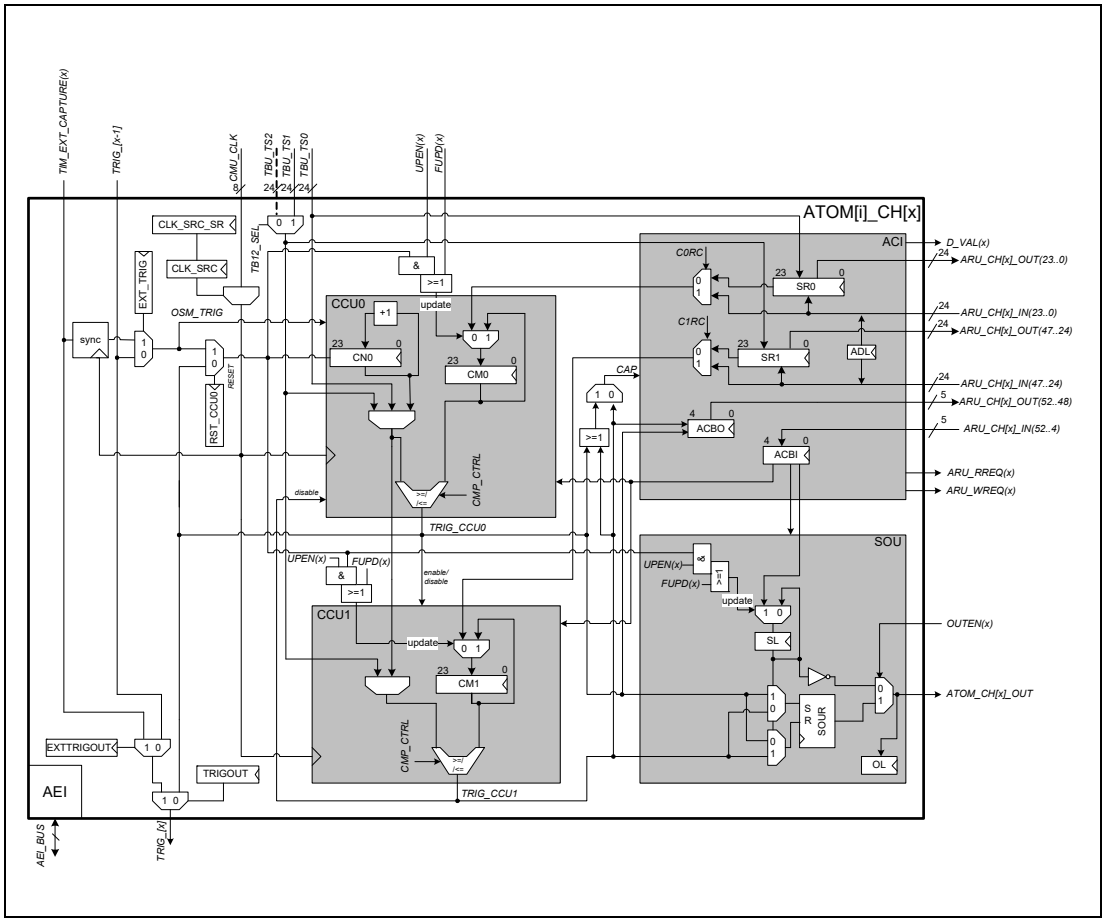



**Figure 25.34 ATOM channel architecture**

In all ATOM channels the operation registers CN0, CM0 and CM1 and the shadow registers SR0 and SR1 are the 24 bit width. The comparators inside CCU0 and CCU1 provide a selectable signed greater/equal or less/equal comparison to compare against the GTM time bases TBU_TS0, TBU_TS1 and, if available, TBU_TS2 . See **Section 25.9, Time Base Unit (TBU)** for further details. For an overview of the implemented TBU submodule version see **Section 25.6.1.1, GTM Architecture Block Diagram** . The CCU0 and CCU1 units have different tasks for the different ATOM channel modes.

The signed compare is used to detect time base overflows and to guarantee, that a compare match event can be set up for the future even when the time base will first overflow and then reach the compare value. Please note, that for a correct behavior of this signed compare, the new compare value must not be specified larger/smaller than half of the range of the total time base value ($7FFFFF_H$).

In SOMC/SOMB mode, the two compare units CCUx can be used in combination to each other. When used in combination, the trigger lines TRIG_CCU0 and TRIG_CCU1 can be used to enable/disable the other compare unit on a match event. See **Section 25.11.3.2, ATOM Signal Output Mode**

**Compare (SOMC)** and **Section 25.11.3.5, ATOM Signal Output Mode Buffered Compare(SOMB)** for further details.

The Signal Output Unit (SOU) generates the output signal for each ATOM channel. This output signal level depends on the ATOM channel mode and on the SL bit of the GTM0ATOMixCTRL register in combination with the two control bits. These two control bits ACB(1) and ACB(0) can either be received via CPU in the ACB register field of the GTM0ATOMixCTRL register or via ARU in the ACBI bit field of the GTM0ATOMixSTAT register.

The SL bit in the GTM0ATOMixCTRL register defines in all modes the operational behavior of the ATOM channel.

When the channel and its output is disabled, the output signal level of the channel is the inverse of the SL bit.

In SOMI, SOMC and SOMB mode the output signal level depends on the SL, ACB0 and ACB1 bits. In SOMP mode the output signal level depends on the two trigger signals TRIG_CCU0 and TRIG_CCU1 since theses two triggers define the PWM timing characteristics and the SL bit defines the level of the duty cycle. In SOMS mode the output signal level is defined by the bit pattern that has to be shifted out by the ATOM channel. The bit pattern is located inside the CM1 register.

The ARU Communication Interface (ACI) subunit is responsible for requesting data routed through ARU to the ATOM channel in SOMI, SOMP, SOMB and SOMS modes, and additionally for providing data to the ARU in SOMC mode.

In SOMC mode the ACI shadow registers have a different behavior and are used as output buffer registers for data send to ARU.

### 25.11.2.2 ARU Communication Interface

The ATOM channels have an ARU Communication Interface (ACI) subunit. This subunit is responsible for data exchange from and to the ARU. This is done with the two implemented registers SR0, SR1, and the ACBI and ACBO bit fields that are part of the GTM0ATOMixSTAT register. The ACI architecture is shown in **Figure 25.35**.

If the ARU_EN bit is set inside the GTM0ATOMixCTRL register, the ATOM channel is enabled by setting the enable bits inside the GTM0ATOMiAGCENDISSTAT register and the CPU hasn't written data not equal to zero into the CM0, CM1, SR0, SR1 register, the ATOM channel will first request data from the ARU before the signal generation starts in SOMP, SOMS, SOMC and SOMB mode.

Note: if in SOMP mode there is data inside the CM0 or SR0 register not equal to '0' the channel counter CN0 will start counting immediately, regardless whether the channel has received ARU data yet.

Note: if in SOMS mode there is data inside the CM0 or SR0 register not equal to '0' the channel will start shifting immediately, regardless whether the channel has received ARU data yet.

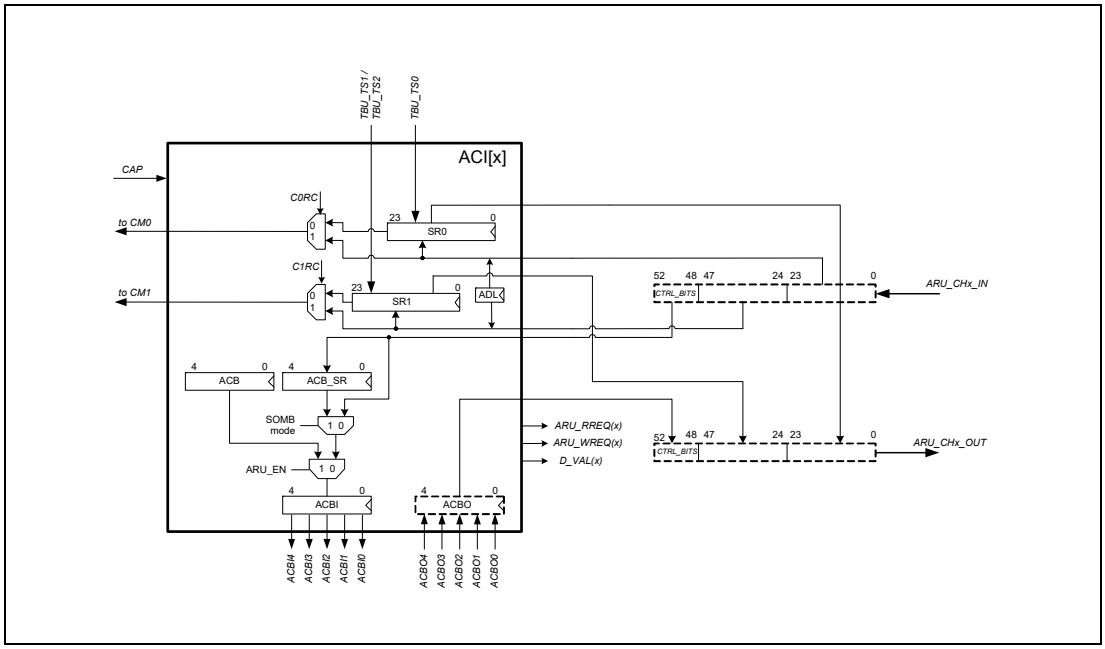



**Figure 25.35 ACI architecture overview**

Incoming ARU data (53 bit width signal ARU_CHx_IN) is split into three parts by the ACI and communicated to the ATOM channel registers. In SOMI, SOMP, SOMS and SOMB modes incoming ARU data ARU_CHx_IN is split in a way that the lower 24 bits of the ARU data (23 down to 0) are stored in the SR0 register, the upper bits (47 down to 24) are stored in the SR1 register. The bits 52 down to 48 (CTRL_BITS) are stored in SOMI, SOMP and SOMS mode in the ACBI bit field of the register GTM0ATOMixSTAT, in SOMB mode in the internal ACB_SR register.

The ATOM channel has to ensure, that in a case when the channel operation registers CM0 and CM1 are updated with the SR0 and SR1 register content and an ARU transfer to these shadow registers happens in parallel that either the old data in both shadow registers is transferred into the operation registers or both new values from the ARU are transferred.

In SOMC mode incoming ARU data ARU_CHx_IN is written directly to the ATOM channel operation register in the way that the lower 24 bits (23 down to 0) are written to CM0, and the bits 47 down to 24 are written to register CM1. The bits 52 down to 48 are stored in the ACBI bit field of the GTM0ATOMixSTAT register and control the behavior of the compare units and the output signal of the ATOM channel.

In SOMC mode the SR0 and SR1 registers serve as capture registers for the time stamps coming from TBU whenever a compare match event is signalled by the CCU0 and/or CCU1 subunits via the CAP signal line. These two time stamps are then provided together with actual ATOM channel status information located in the ACBO bit field to the ARU at the dedicated ARU write address of the ATOM channel when the ARU is enabled.

The encoding of the ARU control bits in the different ATOM operation modes is described in more detail in the following chapters.

### 25.11.3 ATOM Channel modes

As described above, each ATOM channel can operate independently from each other in one of five dedicated output modes:

- ATOM Signal Output Mode Immediate (SOMI)
- ATOM Signal Output Mode Compare (SOMC)
- ATOM Signal Output Mode PWM (SOMP)
- ATOM Signal Output Mode Serial (SOMS)
- ATOM Signal Output Mode Buffered Compare (SOMB)

The Signal Output Mode PWM (SOMP) is principally the same like the output mode for the TOM submodule except the bit reverse mode which is not included in the ATOM. In addition, it is possible to reload the shadow registers over the ARU without the need of a CPU interaction. The three other modes provide additional functionality for signal output control. All operation modes are described in more detail in the following sections.

Note that in any output mode, if a channel is enabled, one-shot mode is disabled (OSM = 0; only used in modes SOMP and SOMS) and CM0 >= CN0, the counter CN0 is incrementing until it reaches CM0.

To avoid unintended counting of CN0 after enabling a channel, it is recommended to reset a channel (or at least CN0 and CM0) before any change on the mode bits MODE, ARU_EN and OSM.

#### 25.11.3.1 ATOM Signal Output Mode Immediate (SOMI)

In ATOM Signal Output Mode Immediate (SOMI), the ATOM channel generates output signals on the ATOM[i]_CH[x]_OUT output port immediate after update of the bit ACBI(0) of register GTM0ATOMixSTAT or ACB(0) bit of register GTM0ATOMixCTRL.

If ARU access is enabled by setting bit ARU_EN in register GTM0ATOMixCTRL, the update of the output ATOM[i]_CH[x]_OUT depends on the bit ACBI(0) of register GTM0ATOMixSTAT received at the ACI subunit and the bit SL bit of register GTM0ATOMixCTRL. The remaining 48 ARU bits (47 down to 0) have no meaning in this mode.

If ARU access is disabled, the update of the output ATOM[i]_CH[x]_OUT depends on the bit ACB(0) and the bit SL of register GTM0ATOMixCTRL.

The initial ATOM channel port pin ATOM[i]_CH[x]_OUT signal level has to be specified by the SL bit field of the GTM0ATOMixCTRL register when OUTEN_CTRL register bit field OUTEN_CTRLx is disabled (see **Section 25.11.6.5, GTM0ATOMiAGCOUTENCTRL (i = 0 to 2)** ) for details.

In SOMI mode the output behavior depends on the SL bit of register GTM0ATOMixCTRL and the bit ACBI(0) of the GTM0ATOMixSTAT register or the bit ACB0 of register GTM0ATOMixCTRL:

**Table 25.91 ATOM Signal Output Mode Immediate (SOMI)**

| SL | ACBI(0)/ACB(0) | Output Behavior |
|---|---|---|
| 0 | 0 | Set output to inverse of SL (1) |
| 0 | 1 | Set output to SL (0) |
| 1 | 0 | Set output to inverse of SL (0) |
| 1 | 1 | Set output to SL (1) |

The signal level bit ACBI(0) is transferred to the SOU subunit of the ATOM and made visible at the output port according to the table above immediately after the data was received by the ACI. This can introduce a jitter on the output signal since the ARU channels are served in a time multiplexed fashion.

### (1) GTM0ATOMixCTRL in SOMI mode (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATM00CTRL: <GTM_base> + $0D004_H$, GTM0ATM01CTRL: <GTM_base> + $0D084_H$, GTM0ATM02CTRL: <GTM_base> + $0D104_H$, GTM0ATM03CTRL: <GTM_base> + $0D184_H$, GTM0ATM04CTRL: <GTM_base> + $0D204_H$, GTM0ATM05CTRL: <GTM_base> + $0D284_H$, GTM0ATM06CTRL: <GTM_base> + $0D304_H$, GTM0ATM07CTRL: <GTM_base> + $0D384_H$, GTM0ATM10CTRL: <GTM_base> + $0D804_H$, GTM0ATM11CTRL: <GTM_base> + $0D884_H$, GTM0ATM12CTRL: <GTM_base> + $0D904_H$, GTM0ATM13CTRL: <GTM_base> + $0D984_H$, GTM0ATM14CTRL: <GTM_base> + $0DA04_H$, GTM0ATM15CTRL: <GTM_base> + $0DA84_H$, GTM0ATM16CTRL: <GTM_base> + $0DB04_H$, GTM0ATM17CTRL: <GTM_base> + $0DB84_H$, GTM0ATM20CTRL: <GTM_base> + $0E004_H$, GTM0ATM21CTRL: <GTM_base> + $0E084_H$, GTM0ATM22CTRL: <GTM_base> + $0E104_H$, GTM0ATM23CTRL: <GTM_base> + $0E184_H$, GTM0ATM24CTRL: <GTM_base> + $0E204_H$

**Value after reset:** 0000 $0x00_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | Not used | — | Not used | | | | | — | — | — | Not used |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | Not used | | | SL | — | — | Not used | | | | ACB(0) | ARU_EN | Not used | MODE | |
| Value after reset | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.92 GTM0ATOMixCTRL in SOMI mode Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 26 | Not used | Not used in this mode. |
| 25 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 24 to 20 | Not used | Not used in this mode. |
| 19 to 17 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 16 | Not used | Not used in this mode. |
| 15 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 14 to 12 | Not used | Not used in this mode. |
| 11 | SL | Initial signal level after channel is enabled.<br>0: Low signal level<br>1: High signal level<br>**NOTES**<br>1. Reset value depends on the hardware configuration chosen by silicon vendor.<br>2. After reset and if channel is disabled, the register SOUR is set to the inverse reset value of bit SL (i.e. '1').<br>If the channel is disabled or the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL. |
| 10, 9 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 8 to 5 | Not used | Not used in this mode. |

**Table 25.92 GTM0ATOMixCTRL in SOMI mode Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 | ACB(0) | ACB bit 0<br>0: Set output to inverse of SL bit<br>1: Set output to SL bit |
| 3 | ARU_EN | ARU Input stream enable<br>0: ARU Input stream disabled<br>1: ARU Input stream enabled |
| 2 | Not used | Not used in this mode. |
| 1, 0 | MODE | ATOM channel mode select.<br>00: ATOM Signal Output Mode Immediate (SOMI) |

### 25.11.3.2 ATOM Signal Output Mode Compare (SOMC)

#### (1) Overview

In ATOM Signal Output Mode Compare (SOMC) the output action is performed in dependence of the comparison between input values located in CM0 and/or CM1 registers and the two (three) time base values TBU_TS0 or TBU_TS1 (or TBU_TS2) provided by the TBU. For a description of the time base generation see the TBU specification in **Section 25.9, Time Base Unit (TBU)**. It is configurable, which of the two (three) time bases is to be compared with one or both values in CM0 and CM1.

The behavior of the two compare units CCU0 and CCU1 is controlled either with the bits 4 down to 2 of ACB bit field inside the GTM0ATOMixCTRL register, when the ARU connection is disabled or with the ACBI bit field of the GTM0ATOMixSTAT register, when the ARU is enabled. In that case the ACB bit field is updated via the ARU control bits 52 down to 48.

The CCUx trigger signals TRIG_CCU0 and TRIG_CCU1 always create edges, dependent on the predefined signal level in SL bit in combination with two control bits that can be specified by either ARU or CPU within the aforementioned GTM0ATOMixCTRL or GTM0ATOMixSTAT registers.

In SOMC mode the channel is always disabled after the specified compare match event occurred. The shadow registers are used to store two time stamp values at the match time. The channel can be enabled again by first reading the shadow registers, either by CPU or ARU and by providing new data for CMx registers through CPU or ARU. For a detailed description see **(2), SOMC Mode under CPU control** and **(3), SOMC Mode under ARU control**.

If three time bases exist for the GTM-IP there must be a preselection between TBU_TS1 and TBU_TS2 for the ATOM channel. This can be done with TB12_SEL bit in the GTM0ATOMixCTRL register.

The comparison in CCU0/1 with time base TBU_TS1 or TBU_TS2 can be done on a greater/equal or less/equal compare according to the CMP_CTRL bit. This control bit has no effect to a compare unit CCU0 or CCU1 that compares against TBU_TS0. In this case always a greater/equal compare is done. The bit CMP_CTRL is part of the GTM0ATOMixCTRL register.

When configured in SOMC mode, the channel port pin has to be initialized to an initial signal level. This initial level after enabling the ATOM channel is determined by the SL bit in the GTM0ATOMixCTRL register. If the output is disabled, the signal level is set to the inverse level of the SL bit.

If the channel is disabled, the register SOUR is set to the SL bit in the GTM0ATOMixCTRL register.

On a compare match event the shadow register SR0 and SR1 are used to capture the TBU time stamp values. SR0 always holds TBU_TS0 and SR1 either holds TBU_TS1 or TBU_TS2 dependent on the TB12_SEL bit in the GTM0ATOMixCTRL register.

Please note, that when the channel is disabled and the compare registers are written, the compare registers CMx are loaded with the written value and the channel starts with the comparison on behalf of this values, when the channel is enabled.

### (2) SOMC Mode under CPU control

As already mentioned above the ATOM channel can be controlled either by CPU or by ARU. When the channel should be controlled by CPU, the ARU_EN bit inside the GTM0ATOMixCTRL register has to be reset.

The output of the ATOM channel is set on a compare match event depending on the ACB10 bit field in combination with the SL bit both located in the GTM0ATOMixCTRL register. The output behavior according to the ACB10 bit field in the control register is shown in the following table:

**Table 25.93 The output behavior according to the ACB10 bit field in the control register**

| SL | ACB10(5) | ACB10(4) | Output behavior |
|---|---|---|---|
| 0 | 0 | 0 | No signal level change at output (exception in **Table 25.96, ATOM CCUx Serve first definition ACB42 = '001'** ) |
| 0 | 0 | 1 | Set output signal level to 1 |
| 0 | 1 | 0 | Set output signal level to 0 |
| 0 | 1 | 1 | Toggle output signal level (exception in **Table 25.96, ATOM CCUx Serve first definition ACB42 = '001'** ) |
| 1 | 0 | 0 | No signal level change at output(exception in **Table 25.96, ATOM CCUx Serve first definition ACB42 = '001'** ) |
| 1 | 0 | 1 | Set output signal level to 0 |
| 1 | 1 | 0 | Set output signal level to 1 |
| 1 | 1 | 1 | Toggle output signal level (exception in **Table 25.96, ATOM CCUx Serve first definition ACB42 = '001'** ) |

The capture/compare strategy of the two CCUx units can be controlled with the ACB42 bit field inside the GTM0ATOMixCTRL register. The meaning of these bits is shown in the following table:

**Table 25.94 The meaning of ACB42 bits (1/2)**

| ACB42(8) | ACB42(7) | ACB42(6) | CCUx control |
|---|---|---|---|
| 0 | 0 | 0 | Serve First: Compare in CCU0 using *TBU_TS0* and in parallel in CCU1 using *TBU_TS1* or *TBU_TS2*. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACB10(5) and ACB10(4).<br>Details see **Table 25.96** |
| 0 | 0 | 1 | Serve First: Compare in CCU0 using *TBU_TS0* and in parallel in CCU1 using *TBU_TS1* or *TBU_TS2*. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACB10(5) and ACB10(4).<br>Details see **Table 25.96** |
| 0 | 1 | 0 | Compare in CCU0 only, use time base *TBU_TS0*. Output signal level is defined by combination of SL, ACB10(5) and ACB10(4) bits. |
| 0 | 1 | 1 | Compare in CCU1 only, use time base *TBU_TS1* or *TBU_TS2*. Output signal level is defined by combination of SL, ACB10(5) and ACB10(4) bits. |
| 1 | 0 | 0 | Serve Last: Compare in CCU0 and then in CCU1 using *TBU_TS0*. Output signal level when CCU0 matches is defined by combination of SL, ACB10(5) and ACB10(4). On the CCU1 match the output level is toggled. |

Table 25.94 The meaning of ACB42 bits (2/2)

| ACB42(8) | ACB42(7) | ACB42(6) | CCUx control |
|---|---|---|---|
| 1 | 0 | 1 | Serve Last: Compare in CCU0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU0 matches is defined by combination of SL, ACB10(5) and ACB10(4). On the CCU1 match the output level is toggled. |
| 1 | 1 | 0 | Serve Last: Compare in CCU0 using TBU_TS0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU1 matches is defined by combination of SL, ACB10(5) and ACB10(4). |
| 1 | 1 | 1 | Not used when ARU disabled. |

The behavior of the ACBI/ACB42 bit combinations ‘000’ and ‘001’ is described in more detail in **Table 25.95, ATOM CCUx Serve first definition ACB42 = ‘000’** and **Table 25.96, ATOM CCUx Serve first definition ACB42 = ‘001’**.

Table 25.95 ATOM CCUx Serve first definition ACB42 = ‘000’

| ACB4 | ACB3 | ACB2 | ACB1 | ACB0 | SL | CCU0 match | CCU1 match | Pin level new |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | hold |
| | | | | | | 1 | 0 | hold |
| | | | | | | 1 | 1 | hold |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| | | | | | | 1 | 0 | 1 |
| | | | | | | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| | | | | | | 1 | 0 | 0 |
| | | | | | | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | toggle |
| | | | | | | 1 | 0 | toggle |
| | | | | | | 1 | 1 | toggle |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | hold |
| | | | | | | 1 | 0 | hold |
| | | | | | | 1 | 1 | hold |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| | | | | | | 1 | 0 | 0 |
| | | | | | | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| | | | | | | 1 | 0 | 1 |
| | | | | | | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | toggle |
| | | | | | | 1 | 0 | toggle |
| | | | | | | 1 | 1 | toggle |

Table 25.96 ATOM CCUx Serve first definition ACB42 = '001'

| ACB4 | ACB3 | ACB2 | ACB1 | ACB0 | SL | CCU0 match | CCU1 match | Pin level new |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | hold |
| | | | | | | 1 | 0 | toggle |
| | | | | | | 1 | 1 | hold |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| | | | | | | 1 | 0 | 1 |
| | | | | | | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| | | | | | | 1 | 0 | 0 |
| | | | | | | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | toggle |
| | | | | | | 1 | 0 | hold |
| | | | | | | 1 | 1 | toggle |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | hold |
| | | | | | | 1 | 0 | toggle |
| | | | | | | 1 | 1 | hold |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| | | | | | | 1 | 0 | 0 |
| | | | | | | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| | | | | | | 1 | 0 | 1 |
| | | | | | | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | toggle |
| | | | | | | 1 | 0 | hold |
| | | | | | | 1 | 1 | toggle |

If the ATOM channel is enabled, the CM0 and/or CM1 registers and the ACB42 bit field of the GTM0ATOMixCTRL register can be updated by the CPU as long as the first match event occurs in case of a serve last compare strategy or as long as the overall match event in case of the other compare strategies.

After a compare match event that causes an update of the shadow registers SR0/SR1 and before reading the SR0 and/or SR1 register via ARU, the update of the registers CM0 and/or CM1 is possible but has no effect.

To set up a new compare action, first the SR0 and/or SR1 register containing captured values have to be read and then new compare values have to be written into the register CM0 and/or CM1.

Which CMx register has to be updated depends on the compare strategy defined in the ACB42 bit field of the channel control register. Since the channel immediately starts with the comparison after the CMx register was/were written, the compare strategy has to be updated before the CMx registers are written.

For the serve last compare strategies, if the register CM0 and CM1 are updated, it can happen that one or both compare values are already located in the past. In any way the ATOM channel will first wait until both compare values are written before it starts the time base comparisons to avoid a deadlock.

The CPU can check at any time if at least one of the ATOM channels' capture compare register contains valid data and waits for a compare event to happen. This is signalled by the DV bit inside the GTM0ATOMixSTAT register.

Note, for serve last compare strategies, if DV bit is currently not set, writing to CM0 or CM1 sets immediately the DV bit although the compare is only started if both values are written.

In SOMC mode and CCUx control mode 'serve last' exist an exception for update of register CM0/CM1. If in this mode the CCU0 compare match event occurred, the update of register CM0/CM1 via CPU is blocked until the CCU1 compare match event.

In the serve last mode (ACB42 = “100” or ACB42 = “101”) it is possible to generate very small spikes on the output pin by loading CM0 and CM1 with two time stamp values for TBU_TS0, TBU_TS1 or TBU_TS2 close together. The output pin will then be set or reset dependent on the SL bit and the specified ACB10(5) and ACB10(4) bits in the ACB10 bit field of the GTM0ATOMixCTRL register on the first match event and the output will toggle on the second compare event in the CCU1 compare unit.

It is important to note, that the bigger (smaller) time stamp has to be loaded into the CM1 register, since the CCU0 will enable the CCU1 once it has reached its comparison time stamp. The order of the comparison time stamps depends on the defined greater/equal or less/equal comparison of the CCUx units.

In addition to storing the captured time stamps in the shadow registers, the ATOM channel provides the result of the compare match event in the ACBO(4) and ACBO(3) bits of the GTM0ATOMixSTAT register. The meaning of the bits is shown in the following table:

**Table 25.97 ACDO bits**

| ACBO(4) | ACBO(3) | Indication |
|---|---|---|
| 0 | 1 | CCU0 compare match occurred |
| 1 | 0 | CCU1 compare match occurred |

Please note, that in case of the ‘serve last’ compare strategy, when the SLA-bit in the GTM0ATOMixCTRL register is not set, the ACBO(4) bit is always set and the ACBO(3) bit is always reset after the compare match event occurred.

The ACBO bit field is reset, when the DV bit is set.

Depending on the capture compare unit where the time base matched the interrupt CCU0TCx_IRQ or CCU1TCx_IRQ is raised.

The behavior of an ATOM channel in SOMC mode under CPU control is visualized in **Figure 25.36** .

**NOTE**

In case of 'serve first' compare strategy, if both events CCU0 and CCU1 occur at the same point in time, both interrupts will be raised.

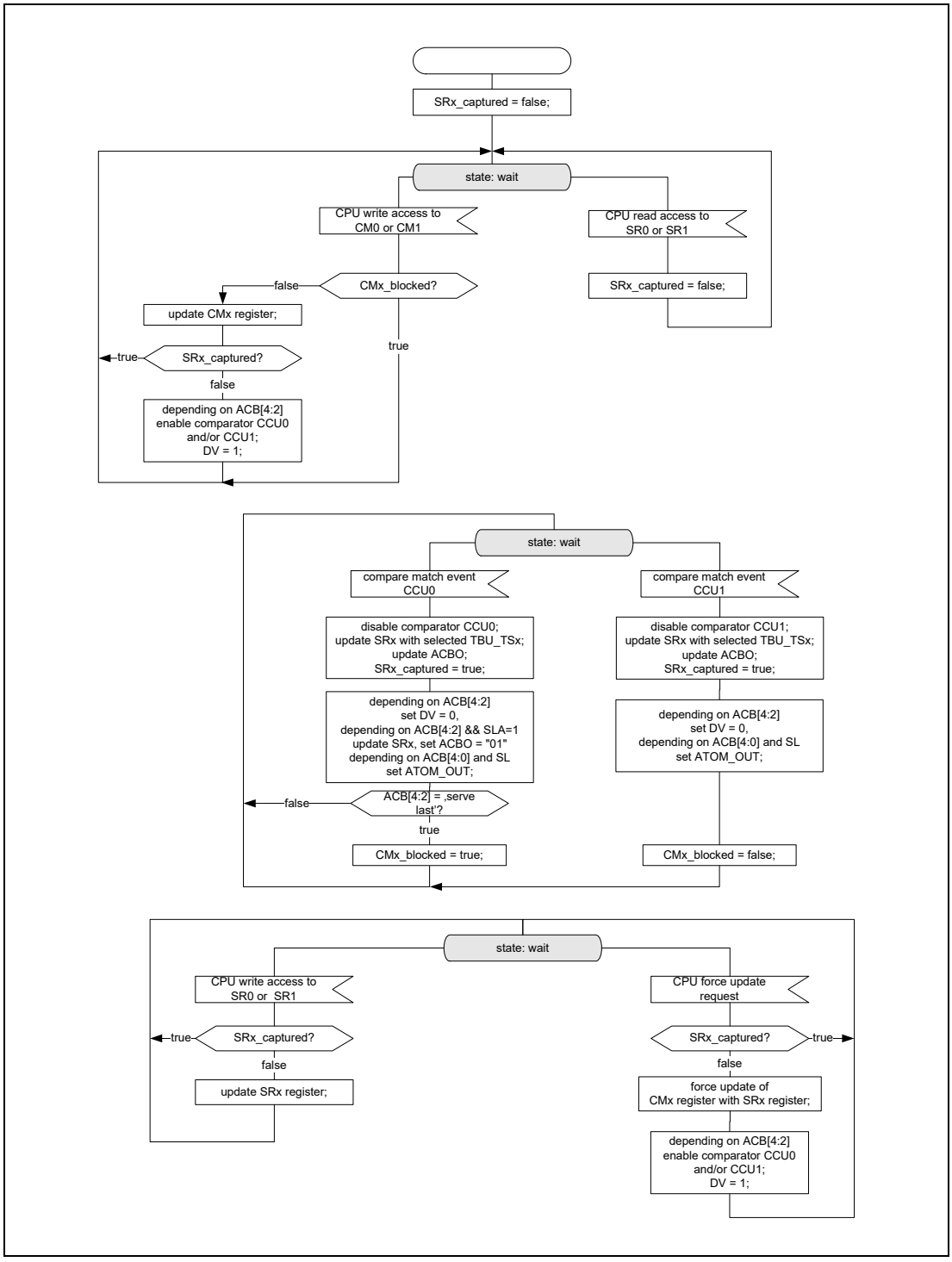



**Figure 25.36 SOMC state diagram for channel under CPU control**

### (3) SOMC Mode under ARU control

When the channel should be controlled by ARU, the ARU_EN bit inside the GTM0ATOMixCTRL register has to be set.

In case, the ATOM channel is under ARU control the content for the compare registers CM0 and CM1 as well as the update of the compare strategy can be loaded via the 53 bit ARU word.

The ARU word 23 to 0 is loaded into the CM0 register while the ARU word 47 to 24 is loaded into the CM1 register. The five ARU control bits 52 to 48 are loaded into the ACBI bit field of the GTM0ATOMixSTAT register and control the channel compare strategy as well as the output behavior in case of compare match events.

For the five ARU control bits 52 to 48 the bits 49 and 48 are loaded into the ACBI bits 1 and 0. The output behavior also depends on the setting of the SL bit inside of the GTM0ATOMixCTRL register and is shown in the following table:

**Table 25.98 The output behavior depends on the setting of the SL bit**

| SL | ACBI(1) | ACBI(0) | Output behavior |
|---|---|---|---|
| 0 | 0 | 0 | No signal level change at output (exception in **Table 25.95** and **Table 25.96**) |
| 0 | 0 | 1 | Set output signal level to 1 |
| 0 | 1 | 0 | Set output signal level to 0 |
| 0 | 1 | 1 | Toggle output signal level (exception in **Table 25.95** and **Table 25.96**) |
| 1 | 0 | 0 | No signal level change at output (exception in **Table 25.95** and **Table 25.96**) |
| 1 | 0 | 1 | Set output signal level to 0 |
| 1 | 1 | 0 | Set output signal level to 1 |
| 1 | 1 | 1 | Toggle output signal level(exception in **Table 25.95** and **Table 25.96**) |

For the five ARU control bits 52 to 48 the bits 52 to 50 are loaded into the ACBI bits 4 to 2. With these three bits the capture/compare units CCUx can be controlled as shown in the following table:

**Table 25.99 CCUxcontrol (1/2)**

| ACBI(4) | ACBI(3) | ACBI(2) | CCUx control |
|---|---|---|---|
| 0 | 0 | 0 | Serve First: Compare in CCU0 using TBU_TS0 and in parallel in CCU1 using TBU_TS1 or TBU_TS2. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACBI(1) and ACBI(0). Details see **Table 25.96** |
| 0 | 0 | 1 | Serve First: Compare in CCU0 using TBU_TS0 and in parallel in CCU1 using TBU_TS1 or TBU_TS2. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACBI(1) and ACBI(0). Details see **Table 25.95** |
| 0 | 1 | 0 | Compare in CCU0 only, use time base TBU_TS0. Output signal level is defined by combination of SL, ACBI(1) and ACBI(0) bits. |
| 0 | 1 | 1 | Compare in CCU1 only, use time base TBU_TS1 or TBU_TS2. Output signal level is defined by combination of SL, ACBI(1) and ACBI(0) bits. |
| 1 | 0 | 0 | Serve Last: Compare in CCU0 and then in CCU1 using TBU_TS0. Output signal level when CCU0 matches is defined by combination of SL, ACBI(1) and ACBI(0). On the CCU1 match the output level is toggled. |

**Table 25.99 CCUxcontrol (2/2)**

| ACBI(4) | ACBI(3) | ACBI(2) | CCUx control |
|---|---|---|---|
| 1 | 0 | 1 | Serve Last: Compare in CCU0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU0 matches is defined by combination of SL, ACBI(1) and ACBI(0). On the CCU1 match the output level is toggled. |
| 1 | 1 | 0 | Serve Last: Compare in CCU0 using TBU_TS0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU1 matches is defined by combination of SL, ACBI(1) and ACBI(0). |
| 1 | 1 | 1 | Change ARU read address to ATOM_RDADDR1<br>DV flag is not set. Neither ACBI(1) nor ACBI(0) is evaluated. |

It is important to note that the bit combination “111” for the ACBI(4), ACBI(3) and ACBI(2) bits forces the channel to request new compare values from another destination read address defined in the ATOM_RDADDR1 bit field of the GTM0ATOMixRDADDR register. After data was successfully received and the compare event occurred the ATOM channel switches back to ATOM_RDADDR0 to receive the next data from there.

After the specified compare match event, the captured time stamps are stored in SR0 and SR1 and the compare result is stored in the ACBO bit field of the GTM0ATOMixSTAT register. The meaning of the ACBO(4) and ACBO(3) bits of the GTM0ATOMixSTAT is shown in the following table:

**Table 25.100 ACBO bits**

| ACBO(4) | ACBO(3) | Return value to ARU |
|---|---|---|
| 0 | 1 | CCU0 compare match occurred |
| 1 | 0 | CCU1 compare match occurred |

Please note, that in case of the ‘serve last’ compare strategy, when the SLA-bit in the GTM0ATOMixCTRL register is not set, the ACBO(4) bit is always set and the ACBO(3) bit is always reset after the compare match event occurred.

The ACBO bit field is reset, when the DV bit is set.

Depending on the capture compare unit where the time base matched the interrupt CCU0TCx_IRQ or CCU1TCx_IRQ is raised.

When CCU0 and CCU1 is used for comparison it is possible to generate very small spikes on the output pin by loading CM0 and CM1 with two time stamp values for TBU_TS0, TBU_TS1 or TBU_TS2 close together. The output pin will then be set or reset dependent on the SL bit and the specified ACBI(0) and ACBI(1) bits in the ACBI bit field of the GTM0ATOMixSTAT register on the first match event and the output will toggle on the second match event.

It is important to note, that the bigger (smaller) time stamp has to be loaded into the CM1 register, since the CCU0 will enable the CCU1 once it has reached its comparison time stamp. The order of the comparison time stamps depends on the defined greater/equal or less/equal comparison of the CCUx units.

For compare strategy ‘serve last’ the CCU0 and CCU1 compare match may occur sequentially. During different phases of compare match the CPU access rights to register CM0 and CM1 as well as to WR_REQ bit is different. These access rights by CPU to register CM0 and CM1 and the WR_REQ are depicted in the following figure.

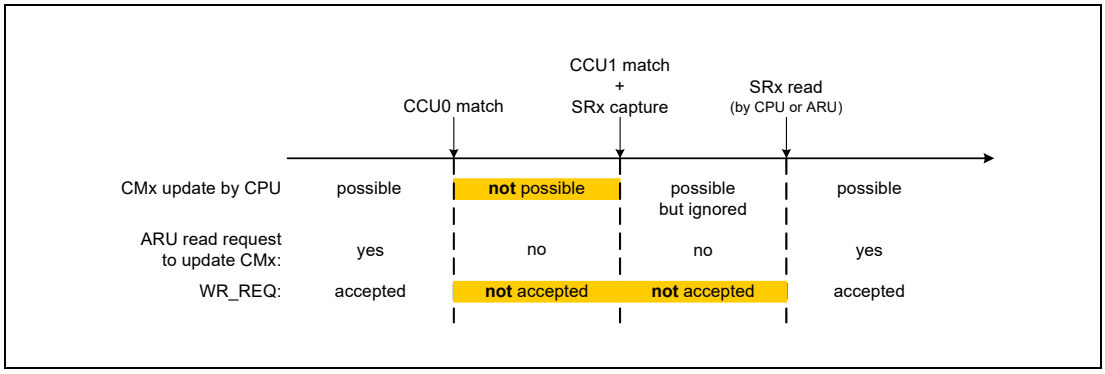



**Figure 25.37 CPU access rights in case of compare strategy 'serve last**

### (a) ARU Non-Blocking mode

When the compare registers are updated via ARU the update behavior of the channel is configurable with the ABM bit inside the GTM0ATOMixCTRL register. When the ABM bit is reset, the ATOM channel is in ARU non-blocking mode.

In this ARU non-blocking mode, data received via ARU is continuously transferred to the registers CM0 and CM1 and the bit field ACBI of register GTM0ATOMixSTAT as long as no specified compare match event occurs.

After a compare match event that causes an update of the shadow register SR0/SR1 and before reading the SR0/SR1 register via CPU or ARU, the update of the registers CM0/ CM1 via CPU or ARU is possible but has no effect.

To set up a new compare action, first the SR0/SR1 registers containing captured values have to be read and then new compare values have to be written into the register CM0/CM1. This can be done either by ARU or by CPU.

When the CPU does the register accesses, only one of the shadow registers has to be read. Dependent on the compare strategy, the CPU has to write one or both of the compare registers.

In SOMC mode and CCUx control mode ‘serve last’ exist an exception for update of register CM0/ CM1. If in this mode the CCU0 compare match event occurred, the update of register CM0/CM1 via CPU or ARU is blocked until the CCU1 compare match event occurs.

The CPU can check at any time if the ATOM channel has received valid data from the ARU and waits for a compare event to happen. This is signalled by the DV bit inside the GTM0ATOMixSTAT register.

The behavior of an ATOM channel in SOMC mode, when ARU is enabled and ARU blocking mode is disabled is shown in **Figure 25.38**.

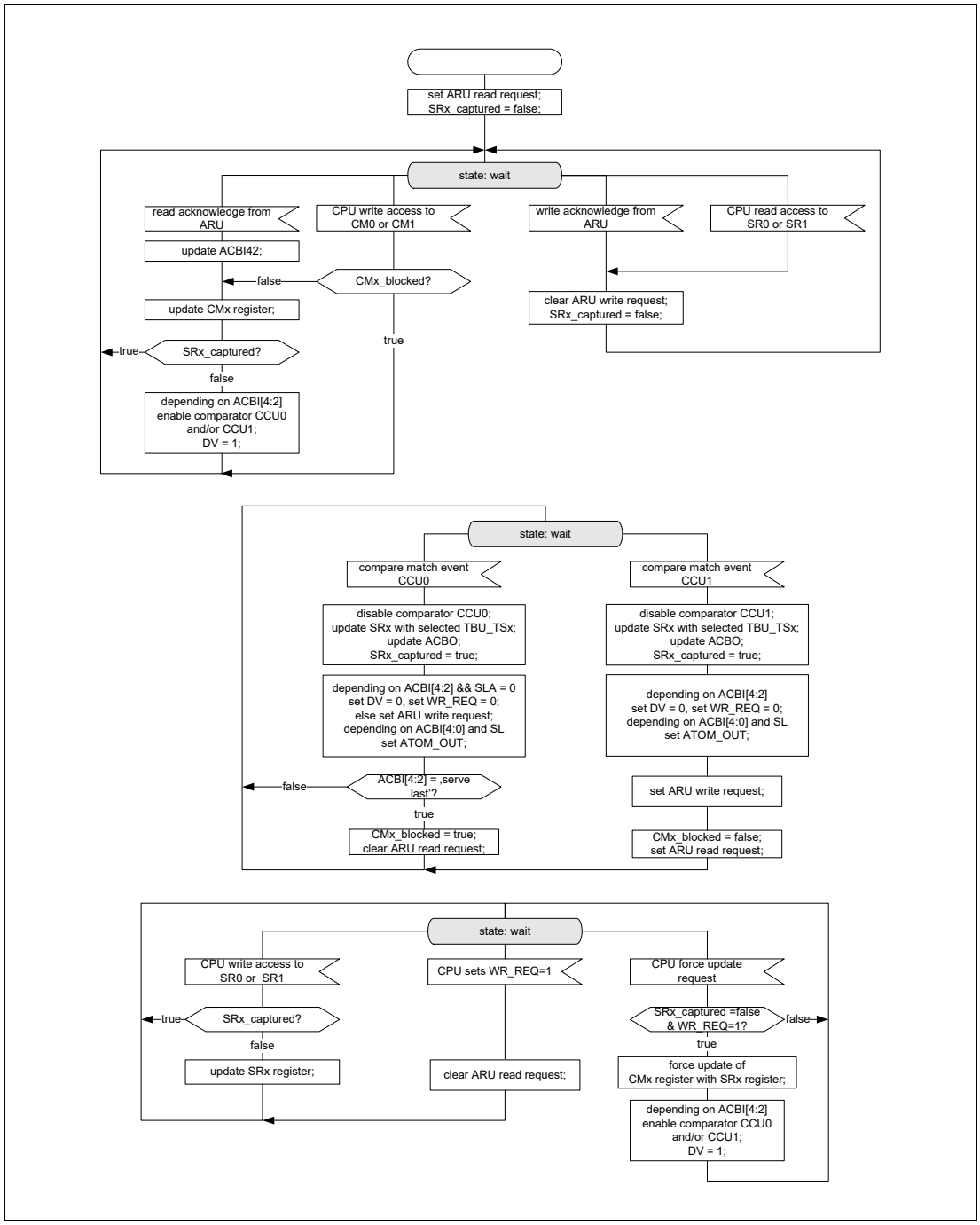



**Figure 25.38 SOMC State diagram for SOMC mode, ARU enabled, ABM disabled**

(b) ARU Blocking mode

When the compare registers are updated by ARU, the ATOM channel can be configured to receive ARU data in a blocking manner. This can be configured by setting the ABM bit in the GTM0ATOMixCTRL register.

If the ABM and ARU_EN bits are set, the (one) two compare values for CM0 and/or CM1 can be provided by ARU or CPU. If the compare registers CM0 and/or CM1 are/is updated, the ATOM channel waits for the compare match event to happen. No further data is requested from the ARU.

When the specified compare match event happens, the shadow registers SR0 and SR1 are updated together with the ACBO bits in the GTM0ATOMixSTAT register. The data in the shadow registers is marked as valid for the ARU and the DV bit is reset inside the GTM0ATOMixCTRL register.

If the register SR0 and SR1 holding the captured TBU time stamp values are read by either the ARU or the CPU, the next write access to or update of the register CM0 or CM1 via ARU or the CPU enables the new compare match check again.

At least one of the registers SR0 or SR1 has to be read, before new data is requested from ARU.

The CPU can check at any time if the ATOM channel has received valid data from the ARU and waits for a compare event to happen. This is signalled by a set DV bit inside the GTM0ATOMixSTAT register.

The behavior of an ATOM channel in SOMC mode, when ARU is enabled and ARU blocking mode is enabled is shown in **Figure 25.39**.

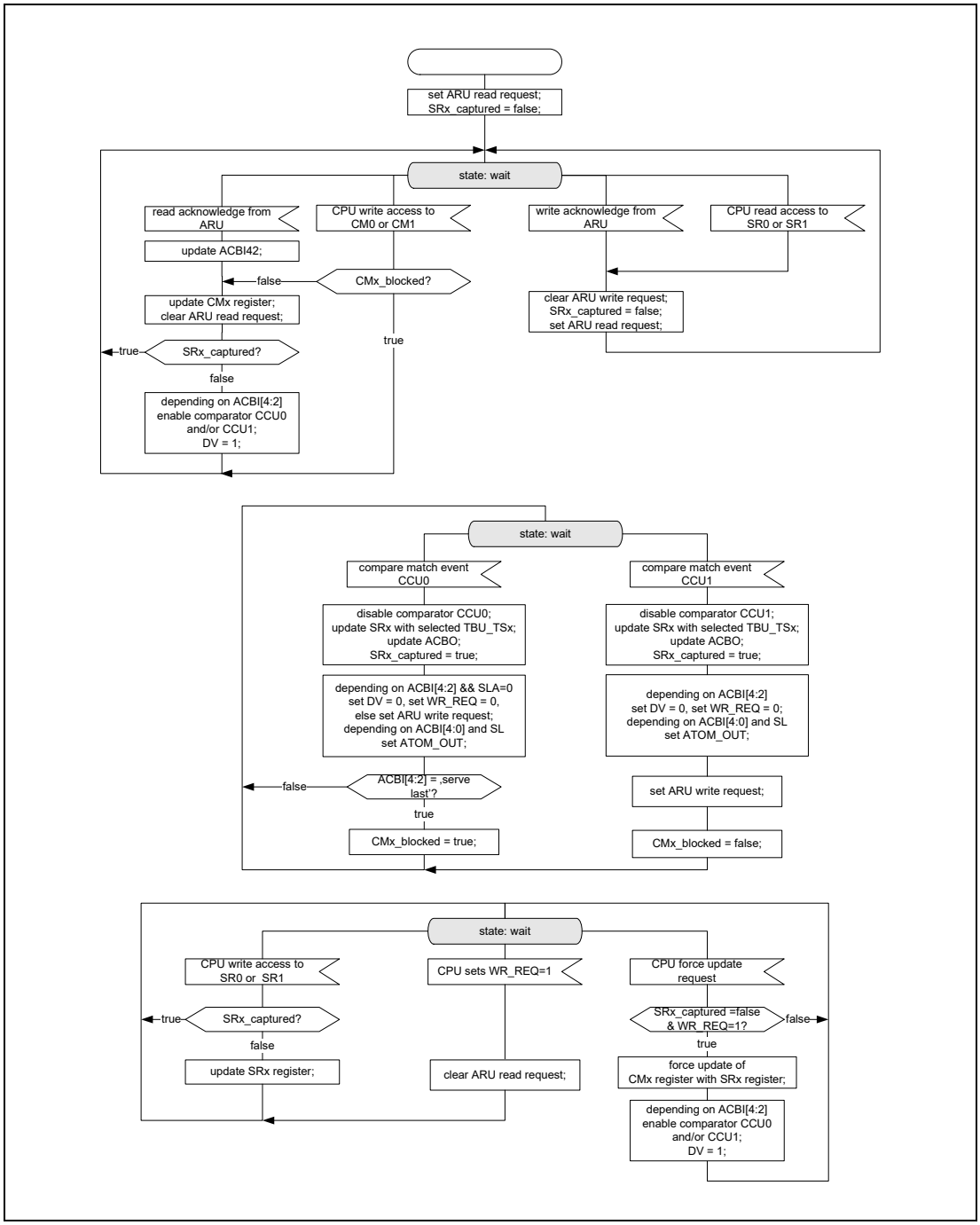



**Figure 25.39 SOMC State diagram for SOMC mode, ARU enabled and ABM enabled**

(c) ATOM SOMC Late update mechanism

Although, the ATOM channel may be controlled by data received via the ARU, the CPU is able to request at any time a late update of the compare register. This can be initiated by setting the WR_REQ bit inside the GTM0ATOMixCTRL register. By doing this, the ATOM will request no further data from ARU (if ARU access was enabled). The channel will in any case continue to compare against the values stored inside the compare registers (if bit DV was set). The CPU can now update the new compare values until the compare event happens by writing to the shadow registers, and force the ATOM channel to update the compare registers by writing to the force update register bits in the AGC register.

If the WR_REQ bit is set and a compare match event happens, any further access to the shadow registers SR0, SR1 is blocked and the force update of this channel is blocked. In addition, the WRF bit is set in the GTM0ATOMixSTAT register. Thus, the CPU can determine that the late update failed by reading the WRF bit.

If a compare match event already happened, the WR_REQ bit could not be set until the channel is unlocked for a new compare match event by reading the shadow registers. In addition, the WRF bit is set if the CPU tries to write the WR_REQ bit in that case.

If between a correct WR_REQ bit set, a correct shadow register write, and before the force update is requested by the AGC a match event occurs on the old compare values, the WRF bit will be set.

The WRF bit will be set in any case if the CPU tries to write to a blocked shadow register.

The WR_REQ bit and the DV bit will be reset on a compare match event.

A blocked force update mechanism will be enabled again after a read access to the register SR0 or SR1 by either the ARU or the CPU.

The ATOM SOMC late update mechanism from CPU is shown in **Figure 25.40**.

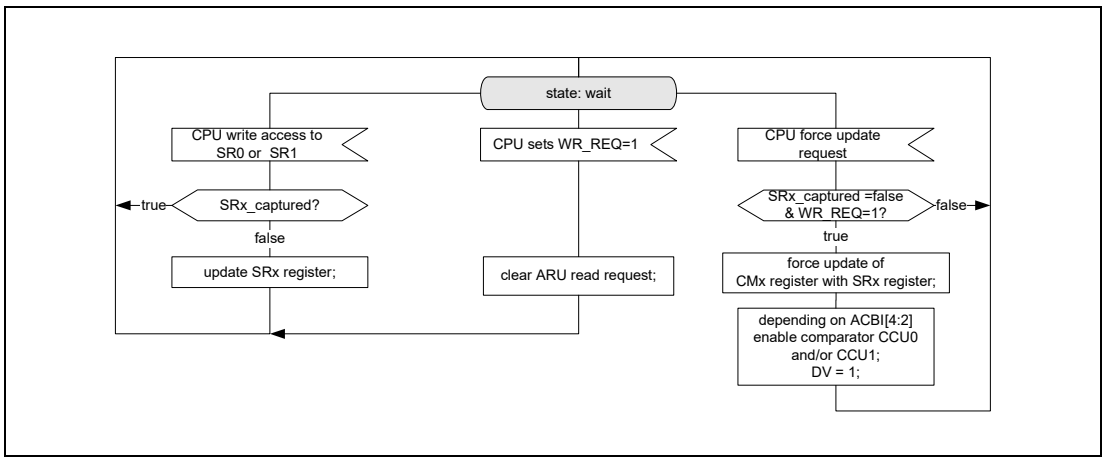



**Figure 25.40 SOMC State diagram for late update requests by CPU**

### (4) Register GTM0ATOMixCTRL in SOMC mode (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0ATM00CTRL: <GTM_base> + $0D004_H$,
GTM0ATM01CTRL: <GTM_base> + $0D084_H$,
GTM0ATM02CTRL: <GTM_base> + $0D104_H$,
GTM0ATM03CTRL: <GTM_base> + $0D184_H$,
GTM0ATM04CTRL: <GTM_base> + $0D204_H$,
GTM0ATM05CTRL: <GTM_base> + $0D284_H$,
GTM0ATM06CTRL: <GTM_base> + $0D304_H$,
GTM0ATM07CTRL: <GTM_base> + $0D384_H$,
GTM0ATM10CTRL: <GTM_base> + $0D804_H$
GTM0ATM11CTRL: <GTM_base> + $0D884_H$
GTM0ATM12CTRL: <GTM_base> + $0D904_H$
GTM0ATM13CTRL: <GTM_base> + $0D984_H$
GTM0ATM14CTRL: <GTM_base> + $0DA04_H$
GTM0ATM15CTRL: <GTM_base> + $0DA84_H$
GTM0ATM16CTRL: <GTM_base> + $0DB04_H$
GTM0ATM17CTRL: <GTM_base> + $0DB84_H$
GTM0ATM20CTRL: <GTM_base> + $0E004_H$
GTM0ATM21CTRL: <GTM_base> + $0E084_H$
GTM0ATM22CTRL: <GTM_base> + $0E104_H$
GTM0ATM23CTRL: <GTM_base> + $0E184_H$
GTM0ATM24CTRL: <GTM_base> + $0E204_H$

**Value after reset:** 0000 $0x00_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ABM | Not used | SLA | TRIGOUT | EXTTRIGOUT | Not used | | Not used | — | — | — | WR_REQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | Not used | | | SL | — | CMP_CTRL | ACB42 | | | ACB10 | | ARU_EN | TB12_SEL | MODE | |
| Value after reset | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.101 GTM0ATOMixCTRL in SOMC mode Register Contents (1/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 27 | ABM | ARU blocking mode<br>0: ARU blocking mode disabled: ATOM reads continuously from ARU and updates CM0, CM1 independent of pending compare match event<br>1: ARU blocking mode enabled: after updating CM0,CM1 via ARU, no new data is read from ARU until compare match event occurred and SR0 and/or SR1 are read. |
| 26 | Not used | Not used in this mode. |

**Table 25.101 GTM0ATOMixCTRL in SOMC mode Register Contents (2/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 25 | SLA | Serve last ARU communication strategy.<br>0: Capture SRx time stamps after CCU0 match event not provided to ARU<br>1: Capture SRx time stamps after CCU0 match event provided to ARU<br>**NOTES**<br>1. Please note, that setting of this bit has only effect, when ACBI(4:2) is configured for serve last compare strategy (“100”, “101”, or “110”).<br>2. When this bit is not set, the captured time stamps in the shadow registers SRx are only provided after the CCU1 match occurred. The ACBO(4:3) bits always return “10” in that case.<br>3. By setting this bit, the ATOM channel also provides the captured time stamps after the CCU0 match event to the ARU. The ACBO(4:3) bits are set to “01” in that case. After the CCU1 match event, the time stamps are captured again in the SRx registers and provided to the ARU. The ACBO(4:3) bits are set to “10”. When the data in the shadow registers after the CCU0 match was not consumed by an ARU destination and the CCU1 match occurs, the data in the shadow registers is overwritten by the new captured time stamps. The ATOM channel does not request new data from the ARU when the CCU0 match values are read from an ARU destination. |
| 24 | TRIGOUT | Trigger output selection (output signal TRIG_CHx) of module ATOM_CHx.<br>0: TRIG_[x] is TRIG_[x-1] or TIM_EXT_CAPTURE(x)<br>1: TRIG_[x] is TRIG_CCU0 |
| 23 | EXTTRIGOUT | Select TIM_EXT_CAPTURE(x) as potential output signal TRIG_[x]<br>0: Signal TRIG_[x-1] is selected as output on TRIG_[x] (if TRIGOUT = 1)<br>1: Signal TIM_EXT_CAPTURE(x) is selected as output on TRIG_[x] (if TRIGOUT = 1) |
| 22, 21 | Not used | Not used in this mode. |
| 20 | Not used | Not used in this mode. |
| 19 to 17 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 16 | WR_REQ | CPU write request bit<br>0: No late update requested by CPU<br>1: Late update requested by CPU<br>**NOTES**<br>1. The CPU can disable subsequent ARU read requests by the channel and can update the shadow registers with new compare values, while the compare units operate on old compare values received by former ARU accesses, if occurred.<br>2. On a compare match event, the WR_REQ bit will be reset by hardware.<br>3. At the point of the force update only the shadow registers SR0 and SR1 are transferred into the CM0, CM1 registers. The output action is still defined by the ACBI bit field described by the ARU together with the old compare values for CM0/CM1. |
| 15 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 14 to 12 | Not used | Not used in this mode. |

**Table 25.101 GTM0ATOMixCTRL in SOMC mode Register Contents (3/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | SL | Initial signal level after channel is enabled.<br>0: Low signal level<br>1: High signal level<br>**NOTES**<br>1. Reset value depends on the hardware configuration chosen by silicon vendor.<br>2. If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL.<br>3. If the channel and output are disabled, in MODE=01 (SOMC mode) the output register of SOU unit is set to value of SL. If the output is enabled afterwards, the output ATOM_OUT[x] is equal to the value of SL. |
| 10 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 9 | CMP_CTRL | CCUx compare strategy select.<br>0: Greater/equal compare against TBU time base values (TBU_TS1/2 ≥ CM0/1)<br>1: Less/equal compare against TBU time base values (TBU_TS1/2 ≤ CM0/1)<br>**NOTE**<br>The compare unit CCU0 or CCU1 that compares against TBU_TS0 (depending on CCUx control mode defined by ACBI(4:2) or ACB42) always performs a greater/equal comparison, independent on CMP_CTRL bit. |
| 8 to 6 | ACB42 | ATOM control bits ACB(4), ACB(3), ACB(2)<br>000: Compare in CCU0 and CCU1 in parallel, disable the CCUx on a compare match on either of compare units. Use TBU_TS0 in CCU0 and TBU_TS1 or TBU_TS2 in CCU1.<br>001: Compare in CCU0 and CCU1 in parallel, disable the CCUx on a compare match on either compare units. Use TBU_TS0 in CCU0 and TBU_TS1 or TBU_TS2 in CCU1.<br>010: Compare in CCU0 only against TBU_TS0.<br>011: Compare in CCU1 only against TBU_TS1 or TBU_TS2.<br>100: Compare first in CCU0 and then in CCU1. Use TBU_TS0.<br>101: Compare first in CCU0 and then in CCU1. Use TBU_TS1 or TBU_TS2.<br>110: Compare first in CCU0 and then in CCU1. Use TBU_TS0 in CCU0 and TBU_TS1 or TBU_TS2 in CCU1.<br>111: Reserved.<br>**NOTE**<br>These bits are only applicable if ARU_EN = '0'. |
| 5, 4 | ACB10 | Signal level control bits.<br>00: No signal level change at output (exception in **Table 25.95** and **Table 25.96** mode ACB42 = 001).<br>01: Set output signal level to 1 when SL bit = 0 else output signal level to 0.<br>10: Set output signal level to 0 when SL bit = 0 else output signal level to 1.<br>11: Toggle output signal level (exception in **Table 25.95** and **Table 25.96** mode ACB42 = 001).<br>**NOTE**<br>These bits are only applicable if ARU_EN = '0'. |
| 3 | ARU_EN | ARU Input stream enable.<br>0: ARU Input stream disabled<br>1: ARU Input stream enabled |

**Table 25.101 GTM0ATOMixCTRL in SOMC mode Register Contents (4/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | TB12_SEL | Select time base value TBU_TS1 or TBU_TS2.<br>0: TBU_TS1 selected for comparison<br>1: TBU_TS2 selected for comparison<br>**NOTE**<br>This bit is only applicable if three time bases are present in the GTM-IP. Otherwise, this bit is reserved. |
| 1, 0 | MODE | ATOM channel mode select.<br>01: ATOM Signal Output Mode Compare (SOMC) |

### 25.11.3.3 ATOM Signal Output Mode PWM (SOMP)

In ATOM Signal Output Mode PWM (SOMP) the ATOM submodule channel is able to generate complex PWM signals with different duty cycles and periods. Duty cycles and periods can be changed synchronously and asynchronously. Synchronous change of the duty cycle and/or period means that the duty cycle or period duration changes after the end of the preceding period. An asynchronous change of period and/or duty cycle means that the duration changes during the actual running PWM period.

The signal level of the pulse generated inside the period can be configured inside the channel control register (SL bit of GTM0ATOMixCTRL register). The initial signal output level for the channel is the reverse pulse level defined by the SL bit. **Figure 25.41** clarifies this behavior.

In SOMP mode, depending on configuration bits RST_CCU0 of register GTM0ATOMixCTRL the counter register CN0 can be reset either when the counter value is equal to the compare value CM0 or when signalled by the ATOM[i] trigger signal TRIG_[x-1] of the preceding channel [x-1] (which can also be the last channel of preceding instance TOM[i-1]) or the trigger signal TIM_EXT_CAPTURE(x) of the assigned TIM channel [x].

In this case, if UPEN_CTRL[x]=1, also the working register CM0, CM1 and CLK_SRC are updated.

Note: As an exception, the input TRIG_[0] of instance ATOM0 is triggered by its own last channel cCATO via signal TRIG_[cCATO]. See **Section 25.18, GTM Device 207** and **Section 25.19, GTM Device 208** for value cCATO of ATOM0.

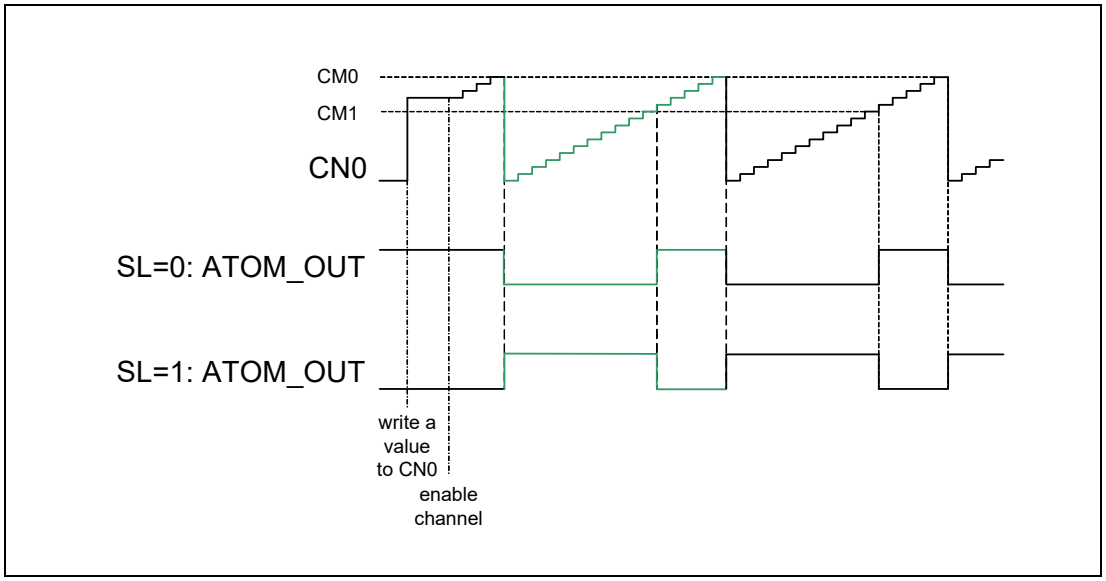



**Figure 25.41 PWM Output behavior with respect to the SL bit in the GTM0ATOMixCTRL register**

On an asynchronous update, it is guaranteed, that no spike occurs at the output port of the channel due to a too late update of the operation registers. The behavior of the output signal due to the different possibilities of an asynchronous update during a PWM period is shown in **Figure 25.42** .

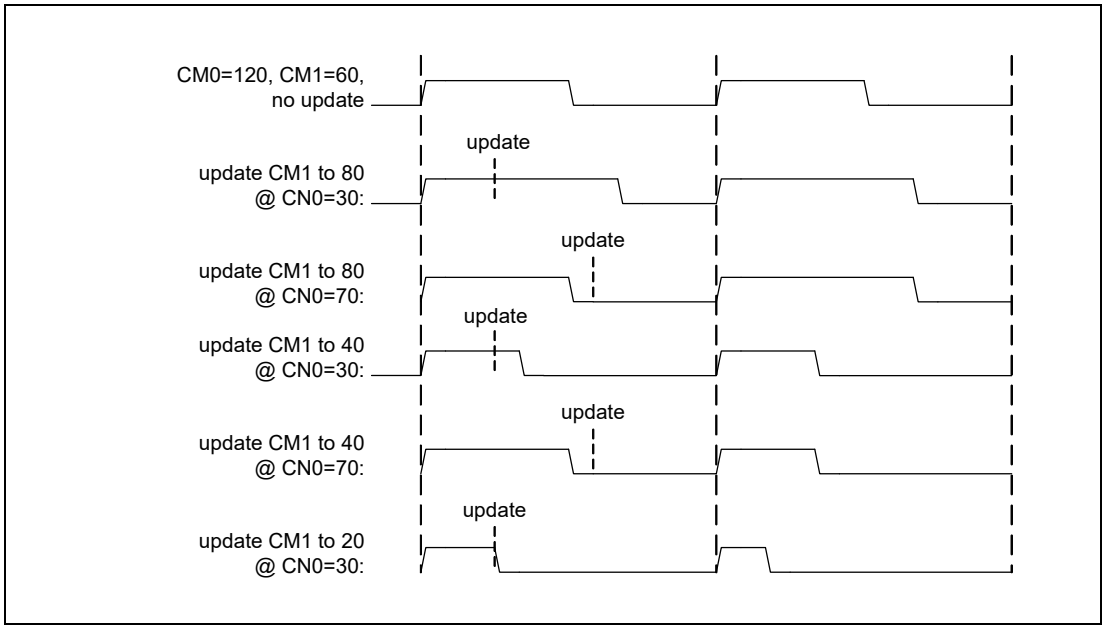



**Figure 25.42 PWM Output behavior in case of an asynchronous update of the duty cycle**

The duration of the pulse high or low time and period is measured with the counter in subunit CCU0. The trigger of the counter is one of the eight CMU clock signals configurable in the channel control register GTM0ATOMixCTRL. The register CM0 holds the duration of the period and the register CM1 holds the duration of the duty cycle in clock ticks of the selected CMU clock.

If counter register CN0 of channel x is reset by its own CCU0 unit (i.e. the compare match of CN0 ≥ CM0 configured by RST_CCU0 = 0), following statements are valid:

- CN0 counts from 0 to CM0-1 and is then reset to 0
- When CN0 is reset from CM0 to 0, an edge to SL is generated.
- When CN0 is incrementing and reaches CN0 > CM1, an edge to !SL is generated.
- if CM0 = 0 or CM0=1, the counter CN0 is constant 0.
- if CM1 = 0, the output is !SL = 0% duty cycle
- if CM1 >= CM0 and CM0>1, the output is SL = 100% duty cycle

If the counter register CN0 of channel x is reset by the trigger signal coming from another channel or the assigned TIM module (configured by RST_CCU0 = 1), following statements are valid:

- CN0 counts from 0 to MAX-1 and is then reset to 0 by trigger signal
- CM0 defines the edge to SL value, CM1 defines the edge to !SL value.
- if CM0 = CM1, the output switches to SL if CN0 = CM0 = CM1 (CM0 has higher priority)
- if CM0 = 0 and CM1= = MAX, the output is SL = 100% duty cycle
- if CM0 > MAX, the output is !SL = 0 % duty cycle, independent of CM1.

In case the counter value CN0 reaches the compare value in register CM0 or the channel receives an external update trigger via the FUPD(x) signal, a synchronous update is performed. A synchronous update means that the registers CM0 and CM1 are updated with the content of the shadow registers SR0 and SR1 and the CLK_SRC register is updated with the value of the CLK_SRC_SR register.

The clock source for the counter can be changed synchronously at the end of a period. If ARU access is disabled, this is done by using the bit field CLK_SRC_SR of register GTM0ATOMixCTRL as shadow registers for the next CMU clock source.

If ARU access is enabled, the bits ACBI(4), ACBI(3) and ACBI(2) received via ARU and stored in register ATOM_[i]_CH[x]_STAT are used as shadow register for the update of the CMU clock source register CLK_SRC.

For the synchronous update mechanism the generation of a complex PWM output waveform is possible without CPU interaction by reloading the shadow registers SR0, SR1 and the ACBI bit field over the ACI subunit from the ARU, while the ATOM channel operates on the CM0 and CM1 registers.

This internal update mechanism is established, when the old PWM period ends. The shadow registers are loaded into the operation registers, the counter register is reset, the new clock source according to the CLK_SRC_SR or ACBI(4), ACBI(3) and ACBI(2) bits is selected and the new PWM generation starts.

In parallel, the ATOM channel issues a read request to the ARU to reload the shadow registers with new values while the ATOM channel operates on the operation registers. To guarantee the reloading, the PWM period must not be smaller than the worst case ARU round trip time and source for the PWM characteristic must provide the new data within this time. Otherwise, the old PWM values are used from the shadow registers.

When updated over the ARU the user has to ensure that the new period duration is located in the lower (bits 23 to 0) and the duty cycle duration is located in the upper (bits 47 to 24) ARU data word and the new clock source is specified in the ARU control bits 52 to 50.

This pipelined data stream character is shown in **Figure 25.43** .

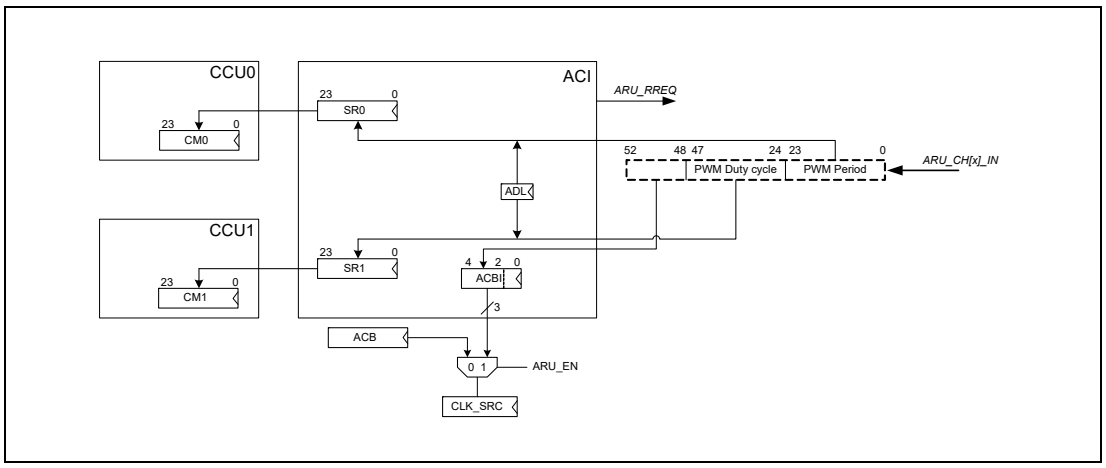



**Figure 25.43 ARU Data input stream pipeline structure for SOMP mode**

When an ARU transfer is in progress which means the ARU_RREQ is served by the ARU, the ACI locks the update mechanism of CM0, CM1 and CLK_SRC until the read request has finished. The CCU0 and CCU1 operate on the old values when the update mechanism is locked.

The shadow registers SR0 and SR1 can also be updated over the AEI bus interface. When updated via the AEI bus the CM0 and CM1 update mechanism has to be locked via the AGC_GLB_CTRL register with the UPENx signal in the AGC subunit. To select the new clock source in this case, the CPU has to write to the CLK_SRC_SR bit field of the GTM0ATOMixCTRL register.

For an asynchronous update of the duty cycle and/or period the new values must be written directly into the compare registers CM0 and/or CM1 while the counter CN0 continues counting. This update can be done only via the AEI bus interface immediately by the CPU or by the FUPD(x) trigger signal triggered from the AGC global trigger logic. Values received through the ARU interface are never loaded asynchronously into the operation registers CM0 and CM1. Therefore, the ATOM channel can generate a PWM signal on the output port pin ATOM[i]_CH[x]_OUT on behalf of the content of the CM0 and CM1 registers, while it receives new PWM values via the ARU interface ACI in its shadow registers.

On a compare match of CN0 and CM0 or CM1 the output signal level of ATOM[i]_CH[x]_OUT is toggled according to the signal level output bit SL in the GTM0ATOMixCTRL register.

Thus, the duty cycle output level can be changed during runtime by writing the new duty cycle level into the SL bit of the channel configuration register. The new signal level becomes active for the next trigger CCU_TRIGx (since bit SL is written).

Since the ATOM[i]_CH[x]_OUT signal level is defined as the reverse duty cycle output level when the ATOM channel is enabled, a PWM period can be shifted earlier by writing an initial offset value to CN0 register. By doing this, the ATOM channel first counts until CN0 reaches CM0 and then it toggles the output signal at ATOM[i]_CH[x]_OUT.

### (1) SOMP One-shot mode

The ATOM channel can operate in One-shot mode when the OSM bit is set in the channel control register. One-shot mode means that a single pulse with the pulse level defined in bit SL is generated on the output line.

First the channel has to be enabled by setting the corresponding ENDIS_STAT value.

In One-shot mode the counter CN0 will not be incremented once the channel is enabled.

A write access to the register CN0 triggers the start of pulse generation (i.e. the increment of the counter register CN0).

If the counter CN0 is reset from CM0 back to zero, the first edge at ATOM[i]_CH[x]_OUT is generated.

To avoid an update of CMx register with content of SRx register at this point in time, the automatic update should be disabled by setting UPEN_CTRL[x] = 00 (in register GTM0ATOMixCTRL)

The second edge is generated if CN0 is greater or equal than CM1 (i.e. CN0 was incremented until it has reached CM1 or CN0 is greater than CM1 after an update of CM1).

If the counter CN0 has reached the value of CM0 a second time, the counter stops.

The new value of CN0 determines the start delay of the first edge. The delay time of the first edge is given by (CM0-CN0) multiplied with period defined by current value of CLK_SRC.

**Figure 25.44** clarifies the pulse generation in SOMP One-shot mode.

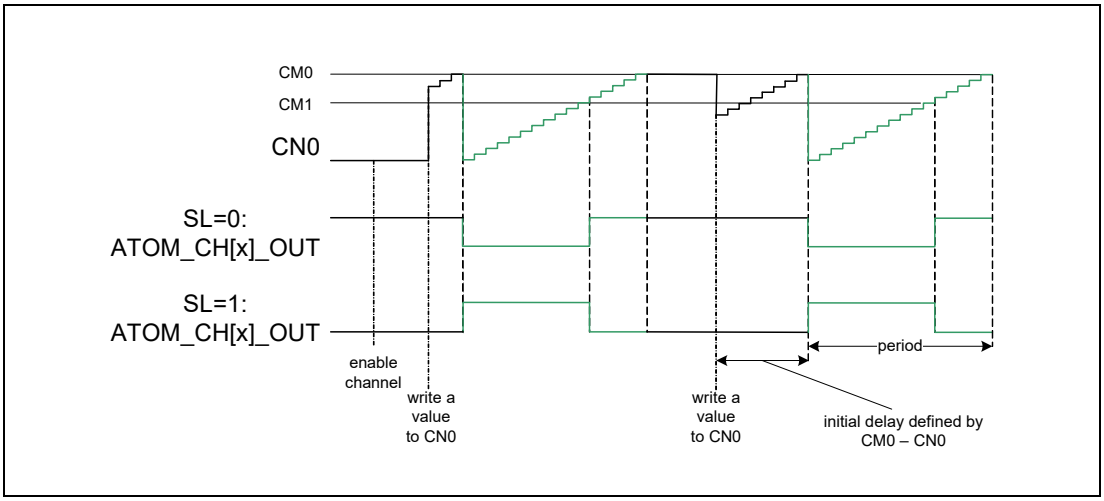



**Figure 25.44 PWM Output with respect to configuration bit SL in One-shot mode: trigger by writing to CN0**

Further output of single pulses can be started by a write access to register CN0.

If CN0 is already incrementing (i.e. started by writing to CN0 a value CN0start < CM0), the affect of a second write access to CN0 depends on the phase of CN0:

phase 1: update of CN0 before CN0 reaches first time CM0

phase 2: update of CN0 after CN0 has reached first time CM0 but is less than CM1

phase 3: update of CN0 after CN0 has reached first time CM0 and CN0 is greater than or equal CM1

In phase 1: writing to counter CN0 a value CN0new < CM0 leads to a shift of first edge (generated if CN0 reaches CM0 first time) by the time CM0-CN0new.

In phase 2: writing to incrementing counter CN0 a value CN0new < CM1 while CN0old is below CM1 leads to a lengthening of the pulse. The counter CN0 stops if it reaches CM0.

In phase 3: Writing to incrementing counter CN0 a value CN0new while CN0old is already greater than or equal CM1 leads to an immediate restart of a single pulse generation inclusive the initial delay defined by CM0 - CN0new.

If a channel is configured to one-shot mode and configuration bit OSM_TRIG is set to 1, the trigger signal OSM_TRIG (i.e. TRIG_[x-1] or TIM_EXT_CAPTURE(x)) triggers start of one pulse generation.

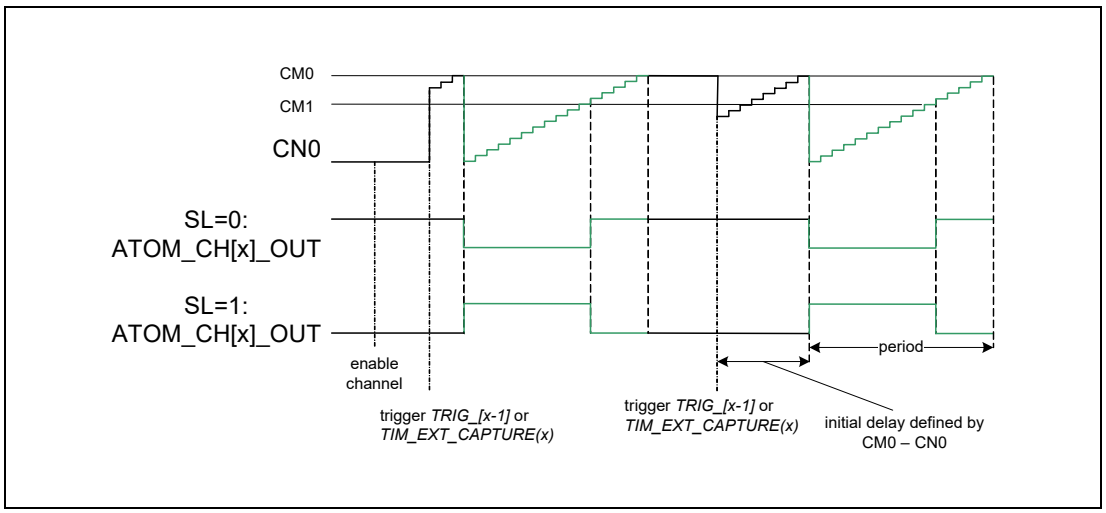



**Figure 25.45 PWM Output with respect to configuration bit SL in one-shot mode: trigger by TRIG_[x-1] orTIM_EXT_CAPTURE(x)**

### (2) PCM mode (pulse count modulation)

At the output ATOM[i]_CH[x]_OUT a pulse count modulated signal can be generated instead of the simple PWM output signal in SOMP mode.

The PCM mode is enabled by setting bit BITREV to 1 (bit 6 in GTM0ATOMixCTRL register).

Please note that it is device specific, in which channel the PCM mode is available. See **Section 25.18, GTM Device 207**and **Section 25.19, GTM Device 208**.

With the configuration bit BITREV = 1 a bit-reversing of the counter output CN0 is configured. In this case the bits LSB and MSB are swapped, the bits LSB+1 and MSB-1 are swapped, the bits LSB+2 and MSB–2 are swapped and so on.

The effect of bit-reversing of the CN0 register value is shown in the following **Figure 25.46**.

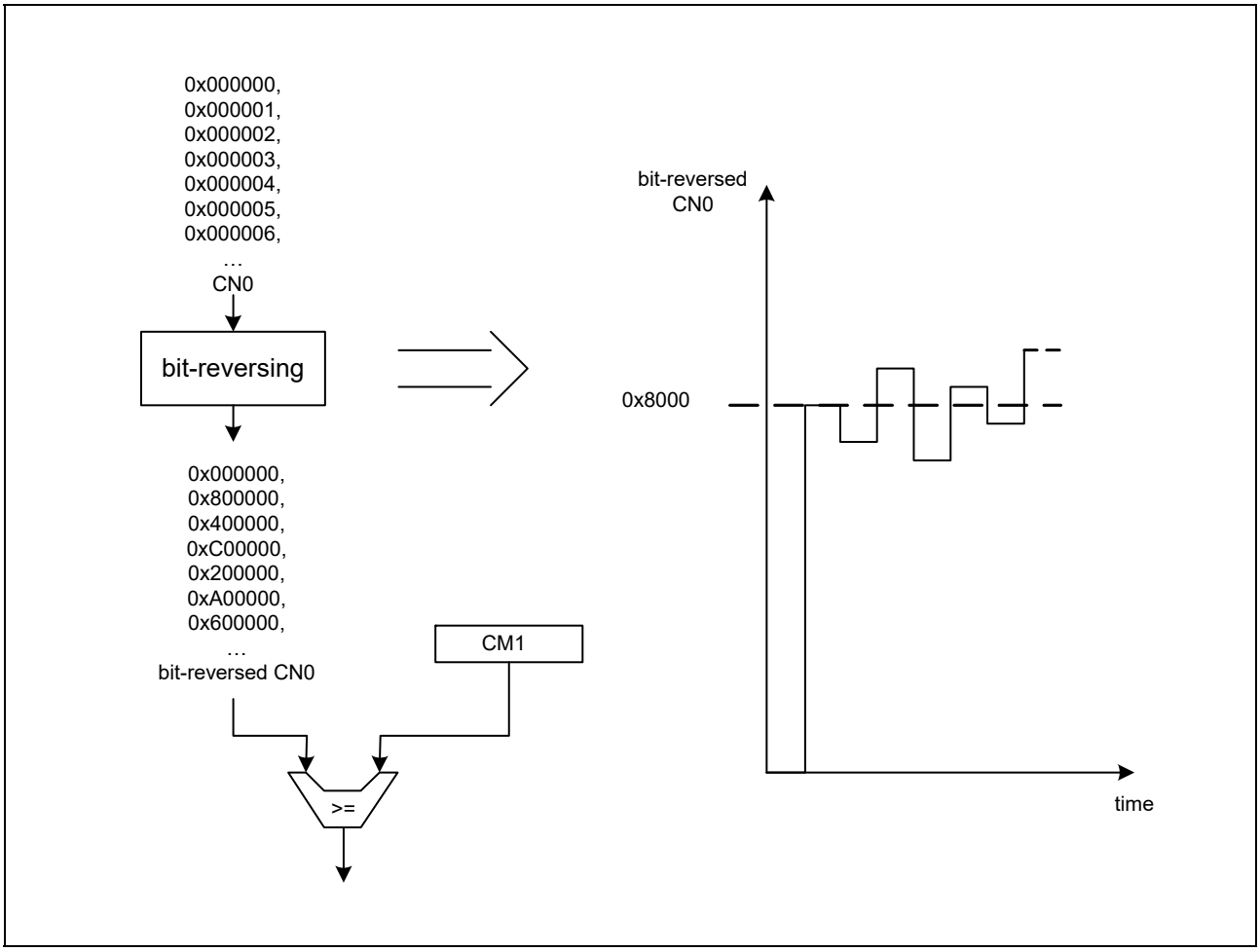



**Figure 25.46 Bit reversing of counter CN0 output**

In the PCM mode the counter register CN0 is incremented by every clock tick depending on configured CMU clock (CMU_CLK).

The output of counter register CN0 is first bit-reversed and then compared with the configured register value CM1.

If the bit-reversed value of register CN0 is greater or equal than CM1, the SR-FlipFlop of submodule SOU is set (i.e. set to inverse value of SL) otherwise the SR-FlipFlop is reset (i.e. to the value of SL). This generates at the output ATOM[i]_CH[x]_OUT a pulse count modulated signal.

In PCM mode the CM0 register - in which the period is defined - normally has to be set to its maximum value $FFFFFF_H$.

To reduce time period of updating duty cycle value in CM1 register, it is additionally possible to setup period value in CM0 register to smaller values than maximum value as described before.

Possible values for CM0 register are each 2 exponentiated with even numbered values e.g. $800000_H$, $400000_H$, $200000_H$ ... .

In this case the duty cycle has to be configured in the following manner.

Depending on how much the period in CM0 register is decreased - means shifted right starting from $1000000_H$ – the duty cycle in CM1 register has to be shifted left (= rotated: shift MSB back into LSB) with same value, e.g. :

period CM0 = $001000_H$ → shifted 8 bits right from $1000000_H$

→ so duty cycle has to be shifted left 8 bit :

e.g.50 % duty cycle = $0008000_H$ → shift 8 bits left → CM1 = $800000_H$

More examples :

| period CM0 | → | duty cycle | → | shift | → | CM1 |
|---|---|---|---|---|---|---|
| $FFFFFF_H$ | → | $800000_H$ | → | no shift | → | $800000_H$ |
| $800000_H$ | → | $400000_H$ | → | shift 1 bit left | → | $800000_H$ |
| $400000_H$ | → | $100000_H$ | → | shift 2 bits left | → | $400000_H$ |
| $200000_H$ | → | $0FFFFF_H$ | → | shift 3 bits left | → | $7FFFF8_H$ |
| $100000_H$ | → | $033333_H$ | → | shift 4 bits left | → | $333330_H$ |
| $080000_H$ | → | $005555_H$ | → | shift 5 bits left | → | $0AAAA0_H$ |
| ... | | | | | | |
| $000020_H$ | → | $000008_H$ | → | shift 19 bits left | → | $400000_H$ |
| $000010_H$ | → | $000005_H$ | → | shift 20 bits left | → | $500000_H$ |
| ... | | | | | | |

**NOTE**

In this mode the interrupt CCU1TC (see register GTM0ATOMixIRQNOTIFY) is set every time if bitreverse value of CN0 is greater or equal than CM1 which may be multiple times during one period. Therefore, from application point of view it is not useful to enable this interrupt.

### (3) GTM0ATOMixCTRL in SOMP mode (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATM00CTRL: <GTM_base> + $0D004_H$, GTM0ATM01CTRL: <GTM_base> + $0D084_H$, GTM0ATM02CTRL: <GTM_base> + $0D104_H$, GTM0ATM03CTRL: <GTM_base> + $0D184_H$, GTM0ATM04CTRL: <GTM_base> + $0D204_H$, GTM0ATM05CTRL: <GTM_base> + $0D284_H$, GTM0ATM06CTRL: <GTM_base> + $0D304_H$, GTM0ATM07CTRL: <GTM_base> + $0D384_H$, GTM0ATM10CTRL: <GTM_base> + $0D804_H$, GTM0ATM11CTRL: <GTM_base> + $0D884_H$, GTM0ATM12CTRL: <GTM_base> + $0D904_H$, GTM0ATM13CTRL: <GTM_base> + $0D984_H$, GTM0ATM14CTRL: <GTM_base> + $0DA04_H$, GTM0ATM15CTRL: <GTM_base> + $0DA84_H$, GTM0ATM16CTRL: <GTM_base> + $0DB04_H$, GTM0ATM17CTRL: <GTM_base> + $0DB84_H$, GTM0ATM20CTRL: <GTM_base> + $0E004_H$, GTM0ATM21CTRL: <GTM_base> + $0E084_H$, GTM0ATM22CTRL: <GTM_base> + $0E104_H$, GTM0ATM23CTRL: <GTM_base> + $0E184_H$, GTM0ATM24CTRL: <GTM_base> + $0E204_H$

**Value after reset:** 0000 $0x00_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | Not used | OSM | — | TRIGOUT | Not used | | | RST_CCU0 | — | — | — | Not used |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CLK_SRC_SR | | | SL | — | — | Not used | | BITREV | ADL | | ARU_EN | Not used | MODE | |
| Value after reset | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.102 GTM0ATOMixCTRL in SOMP mode Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 27 | Not used | Not used in this mode. |
| 26 | OSM | One-shot mode<br>0: Continuous PWM generation after channel enable<br>1: A single pulse is generated |
| 25 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 24 | TRIGOUT | Trigger output selection (output signal TRIG_CHx) of module ATOM_CHx.<br>0: TRIG_[x] is TRIG_[x-1] or TIM_EXT_CAPTURE(x).<br>1: TRIG_[x] is TRIG_CCU0 |
| 23 to 21 | Not used | Not used in this mode. |
| 20 | RST_CCU0 | Reset source of CCU0<br>0: Reset counter register CN0 to 0 on matching comparison with CM0<br>1: Reset counter register CN0 to 0 on trigger TRIG_[x-1] or TIM_EXT_CAPTURE(x) .<br>**NOTE**<br>If RST_CCU0 = 1 and UPEN_CTRLx = 1 are set, TRIG_[x-1] or TIM_EXT_CAPTURE(x) triggers also the update of work register (CM0, CM1 and CLK_SRC). |
| 19 to 17 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 16 | Not used | Not used in this mode. |
| 15 | Reserved | This bit is always read as 0. When written, write the initial value. |

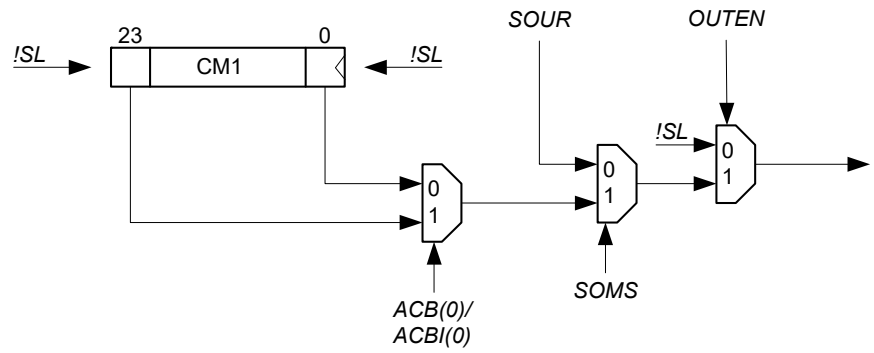
**Table 25.102 GTM0ATOMixCTRL in SOMP mode Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 14 to 12 | CLK_SRC_SR | Shadow register for CMU clock source register CLK_SRC<br>000: CMU_CLK0 selected<br>001: CMU_CLK1 selected<br>010: CMU_CLK2 selected<br>011: CMU_CLK3 selected<br>100: CMU_CLK4 selected<br>101: CMU_CLK5 selected<br>110: CMU_CLK6 selected<br>111: CMU_CLK7 selected<br>**NOTES**<br>1. This register is a shadow register for the CMU_CLKx select. Thus, if the CMU_CLK source for PWM generation should be changed during operation, the old CMU_CLK has to operate until the update of the ATOM channels internal CLK_SRC register by the CLK_SRC_SR content is done either by an end of a period or a FORCE_UPDATE.<br>2. After (channel) reset the selected CLK_SRC value is the SYS_CLK (input of Global Clock Divider). To use one of the CMU_CLKx, it is required to perform a forced update of CLK_SRC with the value of CLK_SRC_SR value before/with enabling the channel.<br>3. These bits are only applicable if ARU_EN = '0'. |
| 11 | SL | Signal level for pulse of PWM.<br>0: Low signal level<br>1: High signal level<br>**NOTE**<br>Reset value depends on the hardware configuration chosen by silicon vendor. If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL. |
| 10, 9 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 8, 7 | Not used | Not used in this mode. |
| 6 | BITREV | Bit-reversing of output of counter register CN0. This bit enables the PCM mode.<br>**NOTE**<br>It is device specific, in which channel the PCM mode is available. See device specific **Section 25.18, GTM Device 207** and **Section 25.19, GTM Device 208** for this information. |
| 5, 4 | ADL | ARU data select for SOMP.<br>00: Load both ARU words into shadow registers<br>01: Load ARU low word (Bits 23 to 0) into shadow register SR0<br>10: Load ARU high word (Bits 47 to 24) into shadow register SR1<br>11: Reserved<br>**NOTE**<br>This bit field is only used in SOMP mode to select the ARU data source. |
| 3 | ARU_EN | ARU Input stream enable.<br>0: ARU Input stream disabled<br>1: ARU Input stream enabled |
| 2 | Not used | Not used in this mode. |
| 1, 0 | MODE | ATOM channel mode select.<br>10: ATOM Signal Output Mode PWM (SOMP) |

### 25.11.3.4 ATOM Signal Output Mode Serial (SOMS)

In ATOM Signal Output Mode Serial (SOMS) the ATOM channel acts as a serial output shift register where the content of the CM1 register in the CCU1 unit is shifted out whenever the unit is triggered by the selected CMU_CLK input clock signal. The shift direction is configurable with the ACB(0) bit inside the GTM0ATOMixCTRL register when ARU is disabled and the ACBI(0) bit inside the GTM0ATOMixSTAT register when ARU is enabled.

The data inside the CM1 register has to be aligned according to the selected shift direction in the ACB(0)/ACBI(0) bit. This means that when a right shift is selected, that the data word has to be aligned to bit 0 of the CM1 register and when a left shift is selected, that the data has to be aligned to bit 23 of the CM1 register.




**Note:** The !SL means inverse value of SL.

**Figure 25.47 SOMS Mode output generation**

**Figure 25.47** shows the output generation in case of SOMS mode is selected.

In SOMS mode CCU0 runs in counter/compare mode and counts the number of bits shifted out so far. The total number of bits that should be shifted is defined as CM0. The total number of bits that are visible at ATOM_OUT is CM0+1.

When the output is disabled the ATOM_OUT is set to the inverse SL bit definition.

When the content of the CM1 register is shifted out, the inverse signal level is shifted into the CM1 register.

When the output is enabled while UPEN_CTRL[x] is disabled, the ATOM_OUT signal level is defined by CM1 bit 0 or 23, dependent on the shift direction defined by ACB(0) or ACBI(0) register setting. **Figure 25.48** should clarify the ATOM channel start-up behavior in this case for right shift. For left shift the CM1 bit 0 in **Figure 25.48** has to be replaced by CM1 bit 23.

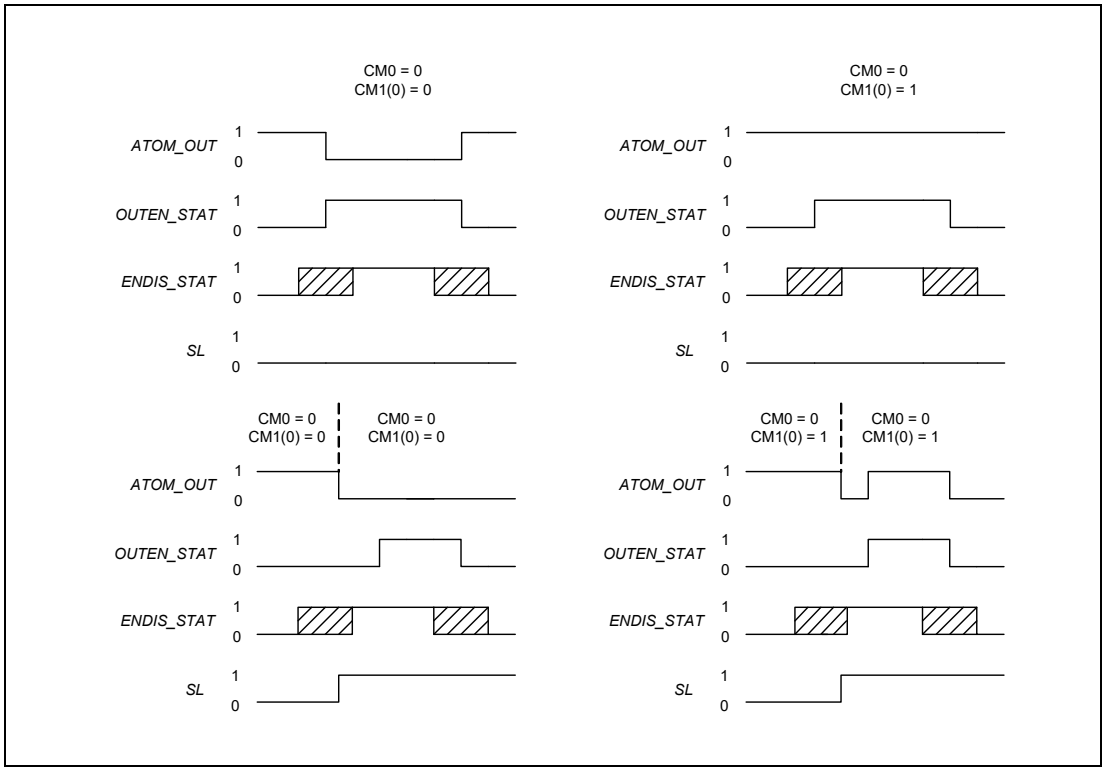



**Figure 25.48 SOMS Output signal level at start-up, UPEN_CTRL[x] disabled**

If UPEN_CTRL[x] is set and the channel is enabled, the output level is defined by bit 0 or 23 of CM1 register dependent on the shift direction. **Figure 25.49** shows the output behavior in that case.

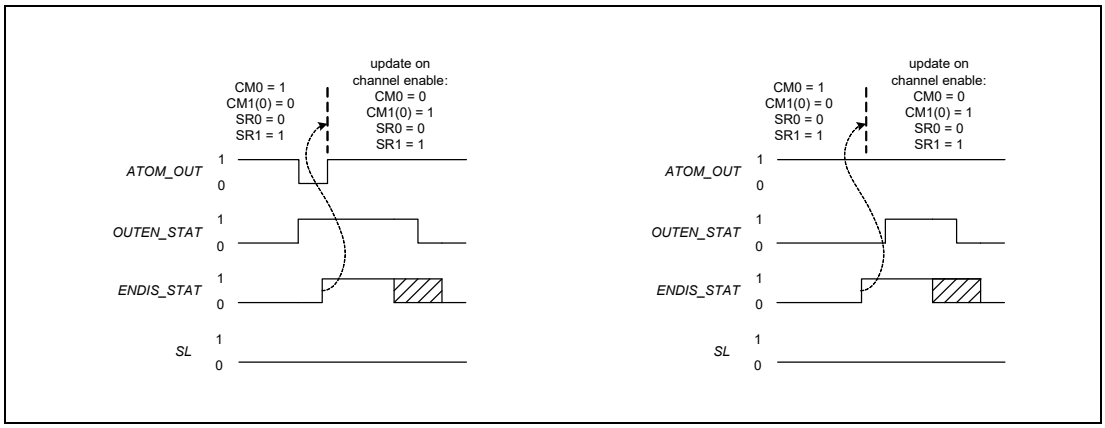



**Figure 25.49 SOMS Output signal level at start-up, UPEN_CTRL[x] enabled**

When the serial data to be shifted is provided via ARU the number of bits that should be shifted has to be defined in the lower 24 bits of the ARU word (23 to 0) and the data that is to be shifted has to be defined in the ARU bits 47 to 24 aligned according to the shift direction. This shift direction has to be defined in the ARU word bit 48 (SL0 bit).

If bit UPEN_CTRL[x] of a channel x is set, after update of CM0/CM1 register with the content of the SR0/SR1 register, a new ARU read request is set up.

If bit UPEN_CTRL[x] of a channel x is not set, no (further) ARU read request is set up (because the SR0/SR1 register are never used for update) and the ATOM may stop shifting after CN0 has reached CM0. Note, that in this case also no automatic restart of shifting is possible.

If a channel is enabled with the settings SOMS mode and ARU_EN = 1, the first received values from ARU are stored in register SR0 and SR1. If CN0 and CM0 are 0 (i.e. CN0 is not counting) and the update of channel x is enabled (UPEN_CTRL[x] = 1), an immediate update of the register CM0 and CM1 is also done. This update of CM0 and CM1 triggers the start of shifting.

It is recommended to configure the ATOM channel in One-shot mode when the ARU_EN bit is not set, since the ATOM channel would reload new values from the shadow registers when CN0 reaches CM0.

**(1) SOMS mode with ARU_EN = 1 and OSM = 0, UPEN_CTRL[x] = 1:**

In case of bit ARU_EN is set and bit OSM is not set, the channel is running in the SOMS continuous mode. Then, if the content of the CM0 register equals the counter CN0, the CM0 and CM1 registers are reloaded with the SR0 and SR1 content and new values are requested from the ARU. If the update of the shadow registers does not happen before CN0 reaches CM0 the old values of SR0 and SR1 are used to reload the operation registers.

In contrast to controlling the channel via AEI, the shift direction defined by ARU word bit 48 has only effect after the update of CMx operation registers from the SRx registers.

**(2) SOMS mode with ARU_EN = 1 and OSM = 1, UPEN_CTRL[x] = 1:**

In case of bit ARU_EN is set and bit OSM is set, the channel is running in the SOMS one-shot mode. Then, if the content of the CM0 register equals the counter CN0 and if new values are available in SR0 and SR1 (bit DV set), the CM0 and CM1 registers are reloaded with the SR0 and SR1 content and new values are requested from the ARU. If no new values are available in SR0 and SR1, the register CM0 and CM1 will not be updated, the counter CN0 stops and the ATOM channel continues to request new data from ARU. A later reception of new ARU data in SR0 and SR1 will immediately force the update of the register CM0 and CM1 and restart the counter CN0.

**(3) SOMS mode with ARU_EN = 0 and OSM = 0, UPEN_CTRL[x] = 1:**

In case of bit ARU_EN is not set and bit OSM is not set, the ATOM channel updates its CM0/CM1 register with the content of the SR0/SR1 register and restarts shifting immediately. The first bit of new CM1 register value will be applied at the output without any gap to the last bit of the previous CM1 register value.

**(4) SOMS mode with ARU_EN = 0 and OSM = 1, UPEN_CTRL[x] = 1:**

In case of bit ARU_EN is not set and bit OSM is set, the ATOM channel stops shifting when CN0 reaches CM0 and no update of CM0 and CM1 is performed.

Then, the shifting of the channel can be restarted again by writing a zero (0) to the CN0 register again. Please note, that the CN0 register should be written with a zero since the CN0 register counts the number of bits shifted out by the ATOM channel.

The writing of a zero to CN0 causes also an immediate update of CM0/CM1 register with the content of SR0/SR1 register.

**(5) Interrupts in SOMS mode**

In ATOM Signal Output Mode Serial only the interrupt CCU0TC (GTM0ATOMixIRQNOTIFY) in case of CN0 ≥ CM0 is generated. The interrupt CCU1TC has no meaning and is not generated.

### (6) GTM0ATOMixCTRL in SOMS mode (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0ATM00CTRL: <GTM_base> + $0D004_H$,
GTM0ATM01CTRL: <GTM_base> + $0D084_H$,
GTM0ATM02CTRL: <GTM_base> + $0D104_H$,
GTM0ATM03CTRL: <GTM_base> + $0D184_H$,
GTM0ATM04CTRL: <GTM_base> + $0D204_H$,
GTM0ATM05CTRL: <GTM_base> + $0D284_H$,
GTM0ATM06CTRL: <GTM_base> + $0D304_H$,
GTM0ATM07CTRL: <GTM_base> + $0D384_H$,
GTM0ATM10CTRL: <GTM_base> + $0D804_H$
GTM0ATM11CTRL: <GTM_base> + $0D884_H$
GTM0ATM12CTRL: <GTM_base> + $0D904_H$
GTM0ATM13CTRL: <GTM_base> + $0D984_H$
GTM0ATM14CTRL: <GTM_base> + $0DA04_H$
GTM0ATM15CTRL: <GTM_base> + $0DA84_H$
GTM0ATM16CTRL: <GTM_base> + $0DB04_H$
GTM0ATM17CTRL: <GTM_base> + $0DB84_H$
GTM0ATM20CTRL: <GTM_base> + $0E004_H$
GTM0ATM21CTRL: <GTM_base> + $0E084_H$
GTM0ATM22CTRL: <GTM_base> + $0E104_H$
GTM0ATM23CTRL: <GTM_base> + $0E184_H$
GTM0ATM24CTRL: <GTM_base> + $0E204_H$

**Value after reset:** 0000 $0x00_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | Not used | OSM | — | Not used | Not used | | | Not used | — | — | — | Not used |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R | R/W | R | R | R | R/W | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CLK_SRC_SR | | | SL | — | | Not used | | | | ACB0 | ARU_EN | Not used | MODE | |
| Value after reset | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.103 GTM0ATOMixCTRL in SOMS mode Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 27 | Not used | Not used in this mode. |
| 26 | OSM | One-shot mode<br>0: Continuous shifting is enabled<br>1: Channel stops, after number of bits defined in CM0 is shifted out |
| 25 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 24 | Not used | Not used in this mode. |
| 23 to 21 | Not used | Not used in this mode. |
| 20 | Not used | Not used in this mode. |
| 19 to 17 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 16 | Not used | Not used in this mode. |
| 15 | Reserved | This bit is always read as 0. When written, write the initial value. |

**Table 25.103 GTM0ATOMixCTRL in SOMS mode Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 14 to 12 | CLK_SRC_SR | Shift frequency select for channel<br>000: CMU_CLK0 selected<br>001: CMU_CLK1 selected<br>010: CMU_CLK2 selected<br>011: CMU_CLK3 selected<br>100: CMU_CLK4 selected<br>101: CMU_CLK5 selected<br>110: CMU_CLK6 selected<br>111: CMU_CLK7 selected<br>**NOTE**<br>This register is a shadow register for the CMU_CLKx select. Thus, if the channel should operate on another CMU_CLK then CMU_CKL0 at the beginning, the different CMU_CLK has to be specified inside this register and the CMU_CLK has to be configured with a FORCE_UPDATE in that case before the channel operation would start. |
| 11 | SL | Defines signal level when channel and output is disable<br>0: High signal level<br>1: Low signal level<br>**NOTES**<br>1. Reset value depends on the hardware configuration chosen by silicon vendor.<br>2. If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL.<br>3. If the output is enabled, the output ATOM_OUT[x] is set to bit 0 or 23 of CM1 register.<br>4. The inverse value of SL is shifted into the CM1 register.<br>5. An enable or disable of the channel x has no effect on ATOM_OUT[x]. |
| 10, 9 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 8 to 5 | Not used | Not used in this mode. |
| 4 | ACB0 | Shift direction for CM1 register<br>0: Right shift of data is started from bit 0 of CM1<br>1: Left shift of data is started from bit 23 of CM1<br>**NOTES**<br>1. The data that has to be shifted out has to be aligned inside the CM1 register according to the defined shift direction.<br>2. This bit is only applicable if ARU_EN = ‘0’.<br>3. If the direction (ACB0) is changed the output ATOM_OUT[x] switches immediately to the other 'first' bit of CM1 (bit 0 if ACB0 = 0, bit 23 if ACB0 = 1). |
| 3 | ARU_EN | ARU Input stream enable.<br>0: ARU Input stream disabled<br>1: ARU Input stream enabled |
| 2 | Not used | Not used in this mode. |
| 1, 0 | MODE | ATOM channel mode select.<br>11: ATOM Signal Output Mode Serial (SOMS) |

### 25.11.3.5 ATOM Signal Output Mode Buffered Compare(SOMB)

#### (1) Overview

In ATOM Signal Output Mode buffered Compare (SOMB) the output action is performed according to the comparison result of the input values located in CM0 and/or CM1 registers and the two (three) time base values TBU_TS0 or TBU_TS1 (or TBU_TS2) provided by the TBU. For a description of the time base generation see the TBU specification in **Section 25.9, Time Base Unit (TBU)**. It is configurable, which of the two (three) time bases is to be compared with one or both values in CM0 and CM1.

The compare strategy of the two compare units CCU0 and CCU1 is controlled by the value of bit field ACBI of register GTM0ATOMixSTAT. This bit field is only readable by CPU. If ARU is disabled, the bit field ACBI can only be updated with the value of bit field ACB of register GTM0ATOMixCTRL. If ARU is enabled, the ACBI bit field can be updated with the value of shadow register ACB_SR which contains a value received via ARU or the value of bit field ACB of register GTM0ATOMixCTRL.

The table below lists all valid control configurations for bit field ACBI of register GTM0ATOMixSTAT.

**Table 25.104 ATOM SOMB compare strategies**

| ACBI(4) | ACBI(3) | ACBI(2) | CCUx control |
|---|---|---|---|
| 0 | 0 | 0 | Reserved. Has no effect. |
| 0 | 0 | 1 | Reserved. Has no effect. |
| 0 | 1 | 0 | Compare in CCU0 only, use time base TBU_TS0. Output signal level is defined by combination of SL, ACB10/ACBI(1..0) bits. |
| 0 | 1 | 1 | Compare in CCU1 only, use time base TBU_TS1 or TBU_TS2. Output signal level is defined by combination of SL, ACBI[1:0] bits. |
| 1 | 0 | 0 | Serve Last: Compare in CCU0 and then in CCU1 using *TBU_TS0*. Output signal level when CCU0 matches is defined by combination of SL, ACBI[1:0] . On the CCU1 match the output level is toggled. |
| 1 | 0 | 1 | Serve Last: Compare in CCU0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU0 matches is defined by combination of SL, ACBI[1:0] . On the CCU1 match the output level is toggled. |
| 1 | 1 | 0 | Serve Last: Compare in CCU0 using TBU_TS0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU1 matches is defined by combination of SL, ACBI[1:0] |
| 1 | 1 | 1 | Cancels pending comparison. |

The CCUx trigger signals TRIG_CCU0 and TRIG_CCU1 creates edges depending on the combination of the predefined signal level in SL bit and the two control bits ACBI[1:0].

In SOMB mode, if ARU access is enabled, the new compare values received via ARU are always stored in the shadow register SR0 and SR1 and the ACB bits are stores in an internal register ACB_SR.

If the scheduled compare matches in CCU0 and/or CCU1 are occurred and the SRx register contain new valid values, the register CM0 and CM1 are updated automatically with the content of the corresponding SRx register, the ACBI bit field is updated with the content of internal ACB_SR register and the DV bit of register GTM0ATOMixSTAT is set. If the SRx register and the CMx register contain no valid value, the compare units are waiting in an idle state.

On a compare match of one of the compare units CCUx units the output ATOM_OUT is set according to combination of ACBI bit 1 down to 0 (in register GTM0ATOMixSTAT) and the SL bit of register GTM0ATOMixCTRL.

**Table 25.105 ATOM SOMB output control by ACBI[1:0] and SL**

| SL | ACBI(1) | ACBI(0) | Output Behavior |
|---|---|---|---|
| 0 | 0 | 0 | No signal level change at output. |
| 0 | 0 | 1 | Set output signal level to 1. |
| 0 | 1 | 0 | Set output signal level to 0. |
| 0 | 1 | 1 | Toggle output signal level. |
| 1 | 0 | 0 | No signal level change at output. |
| 1 | 0 | 1 | Set output signal level to 0. |
| 1 | 1 | 0 | Set output signal level to 1. |
| 1 | 1 | 1 | Toggle output signal level. |

In opposite to SOMC mode no time stamp value of TBU is captured in SRx register.

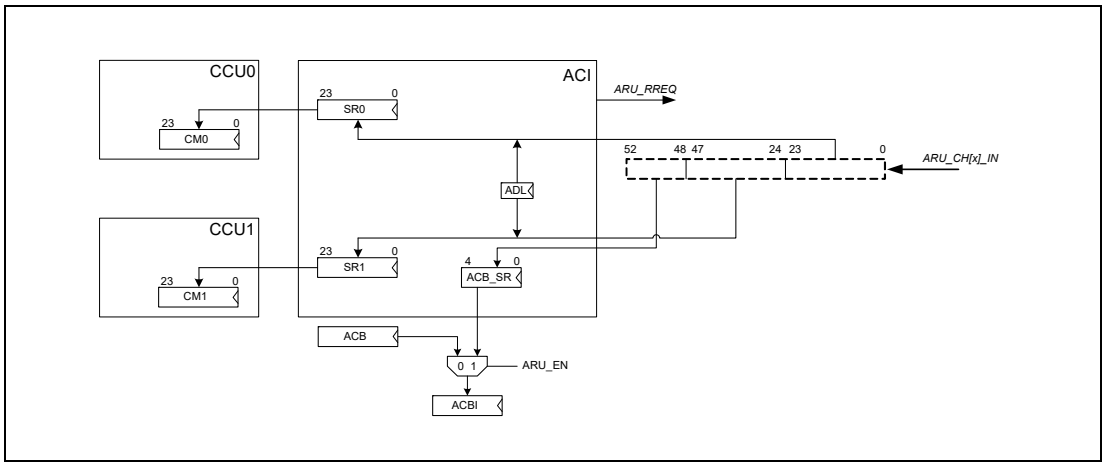



**Figure 25.50 ARU interface behavior in SOMB mode**

The flag DV of register GTM0ATOMixSTAT indicates that at least one of the CMx register contains valid data and a compare event may be pending (if channel is enabled).

The DV flag is reset if none of the CMx register contains valid data.

### (2) SOMB under CPU control

If bit ARU_EN of register GTM0ATOMixCTRL is not set, the ATOM channel can only be controlled via CPU.

Writing to one of the CMx register sets automatically the DV bit to validate the new compare value. A comparison depending on value ACBI of register GTM0ATOMixSTAT is started immediately.

Because only the ACB bit of register GTM0ATOMixCTRL can be written and this bit field serves as a shadow register for the work register ACBI (bit field of register GTM0ATOMixSTAT), it is recommended to first update the ACB bit field before updating CMx/SRx register.

The compare strategy is controlled by the value stored in bit field ACBI of register GTM0ATOMixSTAT. If ARU is disabled, this bit field can only be updated with the value of bit field ACB of register GTM0ATOMixCTRL.

The update of bit field ACBI can be triggered by a forced update or the normal update mechanism controlled by bit UPEN_CTRL[x] in register GTM0ATOMiAGCGLBCTRL.

Writing to one of the SRx register and triggering a forced update, updates the CMx register with the value of SRx register and the ACBI bit field with the content of ACB bit field of register GTM0ATOMiAGCGLBCTRL. A new comparison is started.

Writing to one of the SRx register while update of CMx register is disabled (UPEN_CTRL[x] = 0 in GTM0ATOMiAGCGLBCTRL) and enabling update afterwards, triggers the update of CMx register and the ACBI bit field and starts comparison if previous comparison is finished (DV bit was reset).

If ARU access is disabled (ARU_EN = 0), a force update updates the CMx register with the content of SRx register and the ACBI bit field with the content of ACB bit field of register GTM0ATOMixCTRL.

### (3) SOMB under ARU control

If both compare units CCU0/CCU1 are finished with previous job (depending on compare strategy) and the SRx register contain no new value, they are waiting until new data was received via ARU and stored in SRx register. Then, an immediately update takes place.

If both compare units are finished with previous job (depending on compare strategy) and there are new data available in SRx register, the update the CMx register with the value of the SRx register and the ACBI bit field with the value of internal ACB_SR register takes place and a new compare job is started immediately.

After an update of the CMx register, a new ARU read request is set.

New compare values received via the ARU are stored in shadow register SRx. The ACB bits received via ARU are stored in the internal register ACB_SR.

If ARU access is enabled (ARU_EN = 1), a force update updates the CMx register with the content of SRx register and the ACBI bit field with the content of internal ACB_SR register.

For compare strategy ‘serve last’ the CCU0 and CCU1 compare match may occur sequentially. During different phases of compare match the CPU access rights to register CM0 and CM1 as well as to WR_REQ bit is different. These access rights by CPU to register CM0 and CM1 and the WR_REQ are depicted in the following figure.

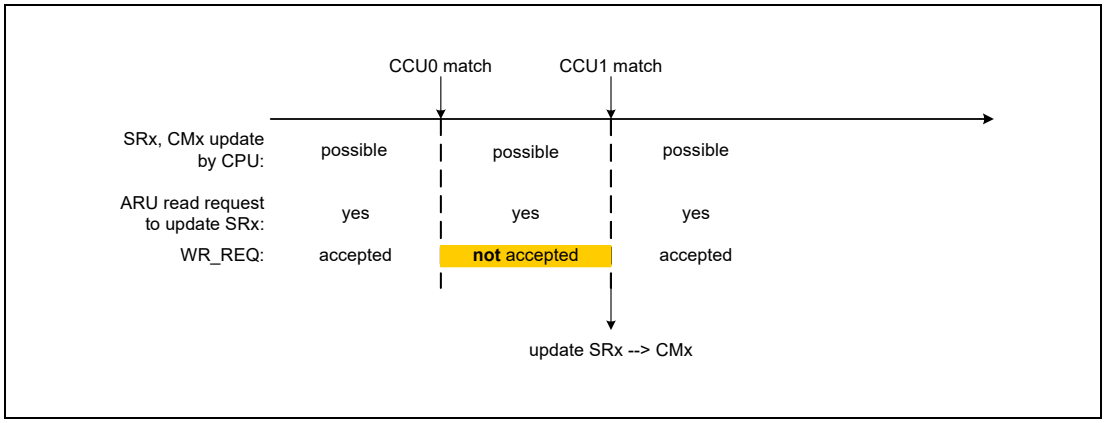



**Figure 25.51 CPU access rights in case of compare strategy ‘serve last’**

(a) ARU Non-blocking mode

If bit ABM in register GTM0ATOMixCTRL is not set, the ARU blocking mode is disabled. In this case the ATOM channel is continuously reading via ARU and storing new values in the SRx register and the ACB shadow register ACB_SR.

If ARU_EN is not set, the bit ABM has no meaning.

(b) ARU Blocking mode

If bit ABM in register GTM0ATOMixCTRL is set, the ARU blocking mode is enabled. In this case the ATOM channel stops requesting new SRx values via ARU after reception of a new SRx value and restarts requesting a new value via ARU after compare match on both compare units (depending on compare strategy) followed by the immediate update of the CMx register with content of SRx register and an update of ACBI with the content of ACB_SR.

If ARU_EN is not set, the bit ABM has no meaning.

(c) Late Update by CPU

Although, the ATOM channel may be controlled by data received via the ARU, the CPU is able to request at any time a late update of the compare register. This can be initiated by setting the WR_REQ bit inside the GTM0ATOMixCTRL register.

If none of the two compare match event happened, the ATOM channel accepts the setting of WR_REQ bit. In this case, the ATOM will request no further data from ARU (if ARU access was enabled) and will disable the update of CMx register with the content of SRx register on a compare match event.

If at least one of the requested compare match events happened (depending on strategy) the WR_REQ bit is not set and the WRF flag in register GTM0ATOMixSTAT is set to indicate that the late update was not successful.

The channel will in any case continue to compare against the values stored inside the compare registers (if bit DV was set). The CPU can now update the compare values by writing to the shadow registers and force the ATOM channel to update the compare registers by writing to the force update register bits in the AGC register.

With a force update the WR_REQ bit is reset automatically and the ARU read request is set up again (if ARU access was enabled).

### (4) GTM0ATOMixCTRL in SOMB mode (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATM00CTRL: <GTM_base> + $0D004_H$, GTM0ATM01CTRL: <GTM_base> + $0D084_H$, GTM0ATM02CTRL: <GTM_base> + $0D104_H$, GTM0ATM03CTRL: <GTM_base> + $0D184_H$, GTM0ATM04CTRL: <GTM_base> + $0D204_H$, GTM0ATM05CTRL: <GTM_base> + $0D284_H$, GTM0ATM06CTRL: <GTM_base> + $0D304_H$, GTM0ATM07CTRL: <GTM_base> + $0D384_H$, GTM0ATM10CTRL: <GTM_base> + $0D804_H$, GTM0ATM11CTRL: <GTM_base> + $0D884_H$, GTM0ATM12CTRL: <GTM_base> + $0D904_H$, GTM0ATM13CTRL: <GTM_base> + $0D984_H$, GTM0ATM14CTRL: <GTM_base> + $0DA04_H$, GTM0ATM15CTRL: <GTM_base> + $0DA84_H$, GTM0ATM16CTRL: <GTM_base> + $0DB04_H$, GTM0ATM17CTRL: <GTM_base> + $0DB84_H$, GTM0ATM20CTRL: <GTM_base> + $0E004_H$, GTM0ATM21CTRL: <GTM_base> + $0E084_H$, GTM0ATM22CTRL: <GTM_base> + $0E104_H$, GTM0ATM23CTRL: <GTM_base> + $0E184_H$, GTM0ATM24CTRL: <GTM_base> + $0E204_H$

**Value after reset:** 0000 $0x00_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | SOMB mode | — | — | ABM | Not used | | TRIGOUT | EXTTRIGOUT | Not used | | Not used | — | — | — | WR_REQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R | R/W | R/W | R/W | R/W | R | R | R | R/W | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | Not used | | | SL | — | CMP_CTRL | ACB[4:2] | | | ACB[1:0] | | ARU_EN | TB12_SEL | MODE | |
| Value after reset | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.106 GTM0ATOMixCTRL in SOMB mode Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 30 | SOMB mode | SOMB mode<br>0: ATOM channel mode defined by bit filed MODE.<br>1: ATOM SOMB mode enabled |
| 29, 28 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 27 | ABM | ARU blocking mode<br>0: ARU blocking mode disabled: ATOM reads continuously from ARU and updates CM0, CM1 independent of pending compare match event.<br>1: ARU blocking mode enabled: after updating CM0,CM1 via ARU, no new data is read from ARU until compare match event occurred and SR0 and/or SR1 are read. |
| 26, 25 | Not used | Not used in this mode. |
| 24 | TRIGOUT | Trigger output selection (output signal TRIG_CHx) of module ATOM_CHx.<br>0: TRIG_[x] is TRIG_[x-1] or TIM_EXT_CAPTURE(x).<br>1: TRIG_[x] is TRIG_CCU0 |
| 23 | EXTTRIGOUT | Select TIM_EXT_CAPTURE(x) as potential output signal TRIG_[x]<br>0: Signal TRIG_[x-1] is selected as output on TRIG_[x] (if TRIGOUT = 1).<br>1: Signal TIM_EXT_CAPTURE(x) is selected as output on TRIG_[x] (if TRIGOUT = 1). |
| 22, 21 | Not used | Not used in this mode. |
| 20 | Not used | Not used in this mode. |
| 19 to 17 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 16 | WR_REQ | CPU Write request bit for late compare register update. |

**Table 25.106 GTM0ATOMixCTRL in SOMB mode Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 14 to 12 | Not used | Not used in this mode. |
| 11 | SL | Initial signal level after channel enable.<br>0: Low signal level<br>1: High signal level<br>**NOTES**<br>1. Reset value depends on the hardware configuration chosen by silicon vendor.<br>2. If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL.<br>3. If the channel and output are disabled, in MODE = 01 (SOMC mode) the output register of SOU unit is set to value of SL. If the output is enabled afterwards, the output ATOM_OUT[x] is equal to the value of SL. |
| 10 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 9 | CMP_CTRL | CCUx compare strategy select.<br>0: Greater/equal compare against TBU time base values (TBU_TS1/2 ≥ CM0/1)<br>1: Less/equal compare against TBU time base values (TBU_TS1/2 ≤ CM0/1)<br>**NOTE**<br>The compare unit CCU0 or CCU1 that compares against TBU_TS0 (depending on CCUx control mode defined by ACB_CM(4:2)) always performs a greater/equal comparison, independent on CMP_CTRL bit. |
| 8 to 6 | ACB[4:2] | ATOM SOMB compare strategy<br>For details see **Table 25.105**<br>000: Reserved. Has no effect.<br>001: Reserved. Has no effect.<br>010: Compare in CCU0 only against TBU_TS0.<br>011: Compare in CCU1 only against TBU_TS1 or TBU_TS2.<br>100: Compare first in CCU0 and then in CCU1. Use TBU_TS0.<br>101: Compare first in CCU0 and then in CCU1. Use TBU_TS1 or TBU_TS2.<br>110: Compare first in CCU0 and then in CCU1. Use TBU_TS0 in CCU0 and TBU_TS1 or TBU_TS2 in CCU1.<br>111: Cancel pending comparisons.<br>**NOTE**<br>These bits are only applicable if ARU_EN = '0'. |
| 5, 4 | ACB[1:0] | Signal level control bits.<br>For details see **Table 25.105**<br>00: No signal level change at output.<br>01: Set output signal level to 1 when SL bit = 0 else output signal level to 0.<br>10: Set output signal level to 0 when SL bit = 0 else output signal level to 1.<br>11: Toggle output signal level.<br>**NOTE**<br>These bits are only applicable if ARU_EN = ‘0’. |
| 3 | ARU_EN | ARU Input stream enable.<br>0: ARU Input stream disabled<br>1: ARU Input stream enabled |

**Table 25.106 GTM0ATOMixCTRL in SOMB mode Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | TB12_SEL | Select time base value TBU_TS1 or TBU_TS2.<br>0: TBU_TS1 selected for comparison<br>1: TBU_TS2 selected for comparison<br>**NOTE**<br>This bit is only applicable if three time bases are present in the GTM-IP. Otherwise, this bit is reserved. |
| 1, 0 | MODE | ATOM channel mode select.<br>Not used in ATOM SOMB mode. |

### 25.11.4 ATOM Interrupt signals

The following table describes ATOM interrupt signals:

**Table 25.107 ATOM interrupt signals**

| Signal | Description |
|---|---|
| CCU0TCx_IRQ | CCU0 Trigger condition interrupt for channel x |
| CCU1TCx_IRQ | CCU1 Trigger condition interrupt for channel x |

### 25.11.5 ATOM Register overview

The following table shows a conclusion of ATOM register address offset and initial values.

**Table 25.108 Register list**

| Symbol | Register Name | Details in Section |
|---|---|---|
| GTM0ATOMiAGCGLBCTRL | AGC Global control register | **25.11.6.1** |
| GTM0ATOMiAGCENDISCTRL | AGC0 Enable/disable control register | **25.11.6.2** |
| GTM0ATOMiAGCENDISSTAT | AGC Enable/disable status register (represents status of ATOM channels) | **25.11.6.3** |
| GTM0ATOMiAGCACTTB | AGC Action time base register | **25.11.6.4** |
| GTM0ATOMiAGCOUTENCTRL | AGC Output enable control register | **25.11.6.5** |
| GTM0ATOMiAGCOUTENSTAT | AGC Output enable status register | **25.11.6.6** |
| GTM0ATOMiAGCFUPDCTRL | AGC Force update control register | **25.11.6.7** |
| GTM0ATOMiAGCINTTRIG | AGC Internal trigger control register | **25.11.6.8** |
| GTM0ATOMixCTRL | ATOM Channel x control register | **(1)** |
| GTM0ATOMixSTAT | ATOM Channel x status register | **(2)** |
| GTM0ATOMixRDADDR | ATOM Channel x ARU read address register | **25.11.6.9** |
| GTM0ATOMixCN0 | ATOM Channel x CCU0 counter register | **25.11.6.10** |
| GTM0ATOMixCM0 | ATOM Channel x CCU0 compare register | **25.11.6.11** |
| GTM0ATOMixSR0 | ATOM Channel x CCU0 compare shadow register | **25.11.6.12** |
| GTM0ATOMixCM1 | ATOM Channel x CCU1 compare register | **25.11.6.13** |
| GTM0ATOMixSR1 | ATOM Channel x CCU1 compare shadow register | **25.11.6.14** |
| GTM0ATOMixIRQNOTIFY | ATOM channel x interrupt notification register | **25.11.6.15** |
| GTM0ATOMixIRQEN | ATOM channel x interrupt enable register | **25.11.6.16** |
| GTM0ATOMixIRQFORCINT | ATOM channel x software interrupt generation | **25.11.6.17** |
| GTM0ATOM00IRQMOD | IRQ mode configuration register | **25.11.6.18** |

## 25.11.6 ATOM Register description

### 25.11.6.1 GTM0ATOMiAGCGLBCTRL (i = 0 to 2)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM0AGCGLBCTRL: <GTM_base> + $0D040_H$
GTM0ATOM1AGCGLBCTRL: <GTM_base> + $0D840_H$
GTM0ATOM2AGCGLBCTRL: <GTM_base> + $0E040_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UPEN_CTRL7 | | UPEN_CTRL6 | | UPEN_CTRL5 | | UPEN_CTRL4 | | UPEN_CTRL3 | | UPEN_CTRL2 | | UPEN_CTRL1 | | UPEN_CTRL0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RST_CH7 | RST_CH6 | RST_CH5 | RST_CH4 | RST_CH3 | RST_CH2 | RST_CH1 | RST_CH0 | — | — | — | — | — | — | — | HOST_TRIG |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R/W |

**Table 25.109 GTM0ATOMiAGCGLBCTRL Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | UPEN_CTRL7 | ATOM channel 7 enable update of register CM0, CM1 and CLK_SRC<br>See bits 17, 16 |
| 29, 28 | UPEN_CTRL6 | ATOM channel 6 enable update of register CM0, CM1 and CLK_SRC<br>See bits 17, 16 |
| 27, 26 | UPEN_CTRL5 | ATOM channel 5 enable update of register CM0, CM1 and CLK_SRC<br>See bits 17, 16 |
| 25, 24 | UPEN_CTRL4 | ATOM channel 4 enable update of register CM0, CM1 and CLK_SRC<br>See bits 17, 16 |
| 23, 22 | UPEN_CTRL3 | ATOM channel 3 enable update of register CM0, CM1 and CLK_SRC<br>See bits 17, 16 |
| 21, 20 | UPEN_CTRL2 | ATOM channel 2 enable update of register CM0, CM1 and CLK_SRC<br>See bits 17, 16 |
| 19, 18 | UPEN_CTRL1 | ATOM channel 1 enable update of register CM0, CM1 and CLK_SRC<br>See bits 17, 16 |
| 17, 16 | UPEN_CTRL0 | ATOM channel 0 enable update of register CM0, CM1 and CLK_SRC from SR0, SR1 and CLK_SRC_SR.<br>Write of following double bit values is possible:<br>00: Don't care, bits 1:0 will not be change<br>01: Update disabled: is read as 00 (see below)<br>10: Update enabled: is read as 11 (see below)<br>11: Don't care, bits 1:0 will not be changed<br>Read of following double values means:<br>00: Channel disabled<br>11: Channel enabled |
| 15 | RST_CH7 | Software reset of channel 7<br>See bit 8 |
| 14 | RST_CH6 | Software reset of channel 6<br>See bit 8 |
| 13 | RST_CH5 | Software reset of channel 5<br>See bit 8 |
| 12 | RST_CH4 | Software reset of channel 4<br>See bit 8 |

**Table 25.109 GTM0ATOMiAGCGLBCTRL Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | RST_CH3 | Software reset of channel 3<br>See bit 8 |
| 10 | RST_CH2 | Software reset of channel 2<br>See bit 8 |
| 9 | RST_CH1 | Software reset of channel 1<br>See bit 8 |
| 8 | RST_CH0 | Software reset of channel 0<br>0: No action<br>1: Reset channel<br>**NOTE**<br>This bit is cleared automatically after write by CPU. The channel registers are set to their reset values and channel operation is stopped immediately. The S-R Flip-Flop SOUR is reset to '1'. |
| 7 to 1 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 0 | HOST_TRIG | Trigger request signal (see AGC) to update the register ENDIS_STAT and OUTEN_STAT<br>0: No trigger request<br>1: Set trigger request<br>**NOTE**<br>This flag is reset automatically after triggering the update |

### 25.11.6.2 GTM0ATOMiAGCENDISCTRL (i = 0 to 2)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM0AGCENDISCTRL: <GTM_base> + $0D044_H$
GTM0ATOM1AGCENDISCTRL: <GTM_base> + $0D844_H$
GTM0ATOM2AGCENDISCTRL: <GTM_base> + $0E044_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENDIS_CTRL7 | | ENDIS_CTRL6 | | ENDIS_CTRL5 | | ENDIS_CTRL4 | | ENDIS_CTRL3 | | ENDIS_CTRL2 | | ENDIS_CTRL1 | | ENDIS_CTRL0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.110 GTM0ATOMiAGCENDISCTRL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15, 14 | ENDIS_CTRL7 | ATOM channel 7 enable/disable update value.<br>See bits 1, 2 |
| 13, 12 | ENDIS_CTRL6 | ATOM channel 6 enable/disable update value.<br>See bits 1, 2 |
| 11, 10 | ENDIS_CTRL5 | ATOM channel 5 enable/disable update value.<br>See bits 1, 2 |
| 9, 8 | ENDIS_CTRL4 | ATOM channel 4 enable/disable update value.<br>See bits 1, 2 |
| 7, 6 | ENDIS_CTRL3 | ATOM channel 3 enable/disable update value.<br>See bits 1, 2 |
| 5, 4 | ENDIS_CTRL2 | ATOM channel 2 enable/disable update value.<br>See bits 1, 2 |
| 3, 2 | ENDIS_CTRL1 | ATOM channel 1 enable/disable update value.<br>See bits 1, 2 |
| 1, 0 | ENDIS_CTRL0 | ATOM channel 0 enable/disable update value.<br>If an ATOM channel is disabled, the counter CN0 is stopped and the Flip-Flop SOUR is set to the inverse value of control bit SL. On an enable event, the counter CN0 starts counting from its current value.<br>Write of following double bit values is possible:<br>00: Don't care, bits 1, 0 of register ENDIS_STAT will not be changed on an update trigger<br>01: Disable channel on an update trigger<br>10: Enable channel on an update trigger<br>11: Don't change bits 1, 0 of this register<br>**NOTE**<br>If the channel is disabled (ENDIS[0] = 0) or the output is disabled (OUTEN[0] = 0), the ATOM channel 0 output ATOM_OUT[0] is the inverted value of bit SL. |

### 25.11.6.3 GTM0ATOMiAGCENDISSTAT (i = 0 to 2)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM0AGCENDISSTAT: <GTM_base> + $0D048_H$
GTM0ATOM1AGCENDISSTAT: <GTM_base> + $0D848_H$
GTM0ATOM2AGCENDISSTAT: <GTM_base> + $0E048_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENDIS_STAT7 | | ENDIS_STAT6 | | ENDIS_STAT5 | | ENDIS_STAT4 | | ENDIS_STAT3 | | ENDIS_STAT2 | | ENDIS_STAT1 | | ENDIS_STAT0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.111 GTM0ATOMiAGCENDISSTAT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15, 14 | ENDIS_STAT7 | ATOM channel 7 enable/disable<br>See bits 1, 2 |
| 13, 12 | ENDIS_STAT6 | ATOM channel 6 enable/disable<br>See bits 1, 2 |
| 11, 10 | ENDIS_STAT5 | ATOM channel 5 enable/disable<br>See bits 1, 2 |
| 9, 8 | ENDIS_STAT4 | ATOM channel 4 enable/disable<br>See bits 1, 2 |
| 7, 6 | ENDIS_STAT3 | ATOM channel 3 enable/disable<br>See bits 1, 2 |
| 5, 4 | ENDIS_STAT2 | ATOM channel 2 enable/disable<br>See bits 1, 2 |
| 3, 2 | ENDIS_STAT1 | ATOM channel 1 enable/disable<br>See bits 1, 2 |
| 1, 0 | ENDIS_STAT0 | ATOM channel 0 enable/disable<br>If an ATOM channel is disabled, the counter CN0 is stopped and the Flip-Flop SOUR is set to the inverse value of control bit SL. On an enable event, the counter CN0 starts counting from its current value.<br>Write of following double bit values is possible:<br>00: Don't care, bits 1, 0 will not be changed<br>01: Channel disabled: is read as 00 (see below)<br>10: Channel enabled: is read as 11 (see below)<br>11: Don't care, bits 1, 0 will not be changed<br>Read of following double values means:<br>00: Channel disable<br>11: Channel enable |

### 25.11.6.4 GTM0ATOMiAGCACTTB (i = 0 to 2)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM0AGCACTTB: <GTM_base> + $0D04C_H$
GTM0ATOM1AGCACTTB: <GTM_base> + $0D84C_H$
GTM0ATOM2AGCACTTB: <GTM_base> + $0E04C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TBU_SEL | | TB_TRIG | ACT_TB | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ACT_TB | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.112 GTM0ATOMiAGCACTTB Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 26, 25 | TBU_SEL | Selection of time base used for comparison<br>00: TBU_TS0 selected<br>01: TBU_TS1 selected<br>10: TBU_TS2 selected<br>11: Same as 00<br>**NOTE**<br>The bit combination “10” is only applicable if the TBU of the device contains three time base channels. Otherwise, this bit combination is also reserved. See GTM Architecture block diagram on page 3 to determine the number of channels for TBU of this device. |
| 24 | TB_TRIG | Set trigger request<br>0: No trigger request<br>1: Set trigger request<br>**NOTE**<br>This flag is reset automatically if the selected time base unit (TBU_TS0 or TBU_TS1 or TBU_TS2 if present) has reached the value ACT_TB and the update of the register were triggered. |
| 23 to 0 | ACT_TB | specifies the signed compare value with selected signal TBU_TS[x], x = 0 to 2. If selected TBU_TS[x] value is in the interval [ACT_TB-007FFFFFh, ACT_TB] the event is in the past and the trigger is generated immediately. Otherwise the event is in the future and the trigger is generated if selected TBU_TS[x] is equal to ACT_TB. |

### 25.11.6.5 GTM0ATOMiAGCOUTENCTRL (i = 0 to 2)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM0AGCOUTENCTRL: <GTM_base> + $0D050_H$
GTM0ATOM1AGCOUTENCTRL: <GTM_base> + $0D850_H$
GTM0ATOM2AGCOUTENCTRL: <GTM_base> + $0E050_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OUTEN_ CTRL7 | | OUTEN_ CTRL6 | | OUTEN_ CTRL5 | | OUTEN_ CTRL4 | | OUTEN_ CTRL3 | | OUTEN_ CTRL2 | | OUTEN_ CTRL1 | | OUTEN_ CTRL0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.113 GTM0ATOMiAGCOUTENCTRL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15, 14 | OUTEN_CTRL7 | Output ATOM_OUT(7) enable/disable update value<br>See bits 1, 2 |
| 13, 12 | OUTEN_CTRL6 | Output ATOM_OUT(6) enable/disable update value<br>See bits 1, 2 |
| 11, 10 | OUTEN_CTRL5 | Output ATOM_OUT(5) enable/disable update value<br>See bits 1, 2 |
| 9, 8 | OUTEN_CTRL4 | Output ATOM_OUT(4) enable/disable update value<br>See bits 1, 2 |
| 7, 6 | OUTEN_CTRL3 | Output ATOM_OUT(3) enable/disable update value<br>See bits 1, 2 |
| 5, 4 | OUTEN_CTRL2 | Output ATOM_OUT(2) enable/disable update value<br>See bits 1, 2 |
| 3, 2 | OUTEN_CTRL1 | Output ATOM_OUT(1) enable/disable update value<br>See bits 1, 2 |
| 1, 0 | OUTEN_CTRL0 | Output ATOM_OUT(0) enable/disable update value<br>Write of following double bit values is possible:<br>00: Don't care, bits 1, 0 of register OUTEN_STAT will not be changed on an update trigger<br>01: Disable channel output on an update trigger<br>10: Enable channel output on an update trigger<br>11: Don't change bits 1, 0 of this register<br>**NOTE**<br>If the channel is disabled (ENDIS[0] = 0) or the output is disabled (OUTEN[0] = 0), the TOM channel 0 output ATOM_OUT[0] is the inverted value of bit SL. |

### 25.11.6.6 GTM0ATOMiAGCOUTENSTAT (i = 0 to 2)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM0AGCOUTENSTAT: <GTM_base> + $0D054_H$
GTM0ATOM1AGCOUTENSTAT: <GTM_base> + $0D854_H$
GTM0ATOM2AGCOUTENSTAT: <GTM_base> + $0E054_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OUTEN_STAT7 | | OUTEN_STAT6 | | OUTEN_STAT5 | | OUTEN_STAT4 | | OUTEN_STAT3 | | OUTEN_STAT2 | | OUTEN_STAT1 | | OUTEN_STAT0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.114 GTM0ATOMiAGCOUTENSTAT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15, 14 | OUTEN_STAT7 | Control/status of output ATOM_OUT(7)<br>See bits 1, 0 |
| 13, 12 | OUTEN_STAT6 | Control/status of output ATOM_OUT(6)<br>See bits 1, 0 |
| 11, 10 | OUTEN_STAT5 | Control/status of output ATOM_OUT(5)<br>See bits 1, 0 |
| 9, 8 | OUTEN_STAT4 | Control/status of output ATOM_OUT(4)<br>See bits 1, 0 |
| 7, 6 | OUTEN_STAT3 | Control/status of output ATOM_OUT(3)<br>See bits 1, 0 |
| 5, 4 | OUTEN_STAT2 | Control/status of output ATOM_OUT(2)<br>See bits 1, 0 |
| 3, 2 | OUTEN_STAT1 | Control/status of output ATOM_OUT(1)<br>See bits 1, 0 |
| 1, 0 | OUTEN_STAT0 | Control/status of output ATOM_OUT(0)<br>Write of following double bit values is possible:<br>00: Don't care, bits 1, 0 will not be changed<br>01: Channel disabled: is read as 00 (see below)<br>10: Channel enabled: is read as 11 (see below)<br>11: Don't care, bits 1, 0 will not be changed<br>Read of following double values means:<br>00: Channel disable<br>11: Channel enable |

### 25.11.6.7 GTM0ATOMiAGCFUPDCTRL (i = 0 to 2)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM0AGCFUPDCTRL: <GTM_base> + $0D058_H$
GTM0ATOM1AGCFUPDCTRL: <GTM_base> + $0D858_H$
GTM0ATOM2AGCFUPDCTRL: <GTM_base> + $0E058_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSTCN0_CH7 | | RSTCN0_CH6 | | RSTCN0_CH5 | | RSTCN0_CH4 | | RSTCN0_CH3 | | RSTCN0_CH2 | | RSTCN0_CH1 | | RSTCN0_CH0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FUPD_CTRL7 | | FUPD_CTRL6 | | FUPD_CTRL5 | | FUPD_CTRL4 | | FUPD_CTRL3 | | FUPD_CTRL2 | | FUPD_CTRL1 | | FUPD_CTRL0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.115 GTM0ATOMiAGCFUPDCTRL Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | RSTCN0_CH7 | Reset CN0 of channel 7 on force update event<br>See bits 17, 16 |
| 29, 28 | RSTCN0_CH6 | Reset CN0 of channel 6 on force update event<br>See bits 17, 16 |
| 27, 26 | RSTCN0_CH5 | Reset CN0 of channel 5 on force update event<br>See bits 17, 16 |
| 25, 24 | RSTCN0_CH4 | Reset CN0 of channel 4 on force update event<br>See bits 17, 16 |
| 23, 22 | RSTCN0_CH3 | Reset CN0 of channel 3 on force update event<br>See bits 17, 16 |
| 21, 20 | RSTCN0_CH2 | Reset CN0 of channel 2 on force update event<br>See bits 17, 16 |
| 19, 18 | RSTCN0_CH1 | Reset CN0 of channel 1 on force update event<br>See bits 17, 16 |
| 17, 16 | RSTCN0_CH0 | Reset CN0 of channel 0 on force update event<br>Write of following double bit values is possible:<br>00: Don't care, bits 1, 0 will not be changed<br>01: CN0 is not reset on forced update: is read as 00 (see below)<br>10: CN0 is reset on forced update: is read as 11 (see below)<br>11: Don't care, bits 1, 0 will not be changed<br>Read of following double values means:<br>00: CN0 is not reset on forced update<br>11: CN0 is reset on forced update |
| 15, 14 | FUPD_CTRL7 | Force update of ATOM channel 7 operation registers<br>See bits 1, 0 |
| 13, 12 | FUPD_CTRL6 | Force update of ATOM channel 6 operation registers<br>See bits 1, 0 |
| 11, 10 | FUPD_CTRL5 | Force update of ATOM channel 5 operation registers<br>See bits 1, 0 |
| 9, 8 | FUPD_CTRL4 | Force update of ATOM channel 4 operation registers<br>See bits 1, 0 |
| 7, 6 | FUPD_CTRL3 | Force update of ATOM channel 3 operation registers<br>See bits 1, 0 |

**Table 25.115 GTM0ATOMiAGCFUPDCTRL Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 5, 4 | FUPD_CTRL2 | Force update of ATOM channel 2 operation registers<br>See bits 1, 0 |
| 3, 2 | FUPD_CTRL1 | Force update of ATOM channel 1 operation registers<br>See bits 1, 0 |
| 1, 0 | FUPD_CTRL0 | Force update of ATOM channel 0 operation registers<br>Write of following double bit values is possible:<br>00: Don't care, bits 1, 0 will not be changed<br>01: Force update disabled: is read as 00 (see below)<br>10: Force update enabled: is read as 11 (see below)<br>11: Don't care, bits 1, 0 will not be changed<br>Read of following double values means:<br>00: Force update disabled<br>11: Force channel enabled |

### 25.11.6.8 GTM0ATOMiAGCINTTRIG (i = 0 to 2)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM0AGCINTTRIG: <GTM_base> + $0D05C_H$
GTM0ATOM1AGCINTTRIG: <GTM_base> + $0D85C_H$
GTM0ATOM2AGCINTTRIG: <GTM_base> + $0E05C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INT_TRIG7 | | INT_TRIG6 | | INT_TRIG5 | | INT_TRIG4 | | INT_TRIG3 | | INT_TRIG2 | | INT_TRIG1 | | INT_TRIG0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.116 GTM0ATOMiAGCINTTRIG Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15, 14 | INT_TRIG7 | Select input signal TRIG_7 as a trigger source<br>See bits 1, 0 |
| 13, 12 | INT_TRIG6 | Select input signal TRIG_6 as a trigger source<br>See bits 1, 0 |
| 11, 10 | INT_TRIG5 | Select input signal TRIG_5 as a trigger source<br>See bits 1, 0 |
| 9, 8 | INT_TRIG4 | Select input signal TRIG_4 as a trigger source<br>See bits 1, 0 |
| 7, 6 | INT_TRIG3 | Select input signal TRIG_3 as a trigger source<br>See bits 1, 0 |
| 5, 4 | INT_TRIG2 | Select input signal TRIG_2 as a trigger source<br>See bits 1, 0 |
| 3, 2 | INT_TRIG1 | Select input signal TRIG_1 as a trigger source<br>See bits 1, 0 |
| 1, 0 | INT_TRIG0 | Select input signal TRIG_0 as a trigger source<br>Write of following double bit values is possible:<br>00: Don't care, bits 1, 0 will not be changed<br>01: Internal trigger from channel 0 (TRIG_0) not used: is read as 00 (see below)<br>10: Internal trigger from channel 0 (TRIG_0) used: is read as 11 (see below)<br>11: Don't care, bits 1, 0 will not be changed<br>Read of following double values means:<br>00: Internal trigger from channel 0 (TRIG_0) not used<br>11: Internal trigger from channel 0 (TRIG_0) used |

### (1) GTM0ATOMixCTRL (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM00CTRL: <GTM_base> + $0D004_H$, GTM0ATOM01CTRL: <GTM_base> + $0D084_H$, GTM0ATOM02CTRL: <GTM_base> + $0D104_H$, GTM0ATOM03CTRL: <GTM_base> + $0D184_H$, GTM0ATOM04CTRL: <GTM_base> + $0D204_H$, GTM0ATOM05CTRL: <GTM_base> + $0D284_H$, GTM0ATOM06CTRL: <GTM_base> + $0D304_H$, GTM0ATOM07CTRL: <GTM_base> + $0D384_H$, GTM0ATOM10CTRL: <GTM_base> + $0D804_H$, GTM0ATOM11CTRL: <GTM_base> + $0D884_H$, GTM0ATOM12CTRL: <GTM_base> + $0D904_H$, GTM0ATOM13CTRL: <GTM_base> + $0D984_H$, GTM0ATOM14CTRL: <GTM_base> + $0DA04_H$, GTM0ATOM15CTRL: <GTM_base> + $0DA84_H$, GTM0ATOM16CTRL: <GTM_base> + $0DB04_H$, GTM0ATOM17CTRL: <GTM_base> + $0DB84_H$, GTM0ATOM20CTRL: <GTM_base> + $0E004_H$, GTM0ATOM21CTRL: <GTM_base> + $0E084_H$, GTM0ATOM22CTRL: <GTM_base> + $0E104_H$, GTM0ATOM23CTRL: <GTM_base> + $0E184_H$, GTM0ATOM24CTRL: <GTM_base> + $0E204_H$

**Value after reset:** 0000 $0X00_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | SOMB | — | — | ABM | OSM | SLA | TRIGOUT | EXTTRIGOUT | EXT_TRIG | OSM_TRIG | RST_CCU0 | — | — | — | WR_REQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CLK_SRC_SR | | | SL | — | CMP_CTRL | ACB | | | | | ARU_EN | TB12_SEL | MODE | |
| Value after reset | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.117 GTM0ATOMixCTRL Register Contents (1/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 30 | SOMB | SOMB mode<br>0: ATOM channel mode defined by bit field MODE<br>1: ATOM SOMB mode enabled |
| 29, 28 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 27 | ABM | ARU blocking mode<br>0: ARU blocking mode disabled: ATOM reads continuously from ARU and updates CM0, CM1 independent of pending compare match event<br>1: ARU blocking mode enabled: after updating CM0,CM1 via ARU, no new data is read from ARU until compare match event is occurred.<br>**NOTE**<br>This bit is only applicable in SOMC mode. |
| 26 | OSM | One-shot mode<br>0: Continuous PWM generation after channel enable<br>1: A single pulse is generated<br>**NOTE**<br>This bit is only applicable in SOMP and SOMS modes. |

**Table 25.117 GTM0ATOMixCTRL Register Contents (2/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 25 | SLA | Serve last ARU communication strategy<br>0: Capture SRx time stamps after CCU0 match event not provided to ARU<br>1: Capture SRx time stamps after CCU0 match event provided to ARU<br>**NOTES**<br>1. This bit is only applicable in SOMC mode.<br>2. Please note, that setting of this bit has only effect, when ACBI(4:2) is configured for serve last compare strategy (“100”, “101”, or “110”).<br>3. When this bit is not set, the captured time stamps in the shadow registers SRx are only provided after the CCU1 match occurred. The ACBO(4:3) bits always return "10" in that case.<br>4. By setting this bit, the ATOM channel also provides the captured time stamps after the CCU0 match event to the ARU. The ACBO(4:3) bits are set to “01” in that case. After the CCU1 match event, the time stamps are captured again in the SRx registers and provided to the ARU. The ACBO(4:3) bits are set to “10”. When the data in the shadow registers after the CCU0 match was not consumed by an ARU destination and the CCU1 match occurs, the data in the shadow registers is overwritten by the new captured time stamps. The ATOM channel does not request new data from the ARU when the CCU0 match values are read from an ARU destination. |
| 24 | TRIGOUT | Trigger output selection (output signal TRIG_CHx) of module ATOM_CHx.<br>0: TRIG_[x] is TRIG_[x-1] orTIM_EXT_CAPTURE(x).<br>1: TRIG_[x] is TRIG_CCU0 |
| 23 | EXTTRIGOUT | Select TIM_EXT_CAPTURE(x) as potential output signal TRIG_[x]<br>0: Signal TRIG_[x-1] is selected as output on TRIG_[x] (if TRIGOUT = 1)<br>1: Signal TIM_EXT_CAPTURE(x) is selected as output on TRIG_[x] (if TRIGOUT = 1) |
| 22 | EXT_TRIG | Select TIM_EXT_CAPTURE(x) as trigger signal<br>0: Signal TIM_[x-1] is selected as trigger to reset CN0 or to start single pulse generation.<br>1: Signal TIM_EXT_CAPTURE(x) is selected |
| 21 | OSM_TRIG | Enable trigger of one-shot pulse by trigger signal *OSM_TRIG*<br>0: Signal OSM_TRIG can not trigger start of single pulse generation<br>1: Signal OSM_TRIG can trigger start of single pulse generation (only if bit OSM = 1)<br>**NOTE**<br>This bit should only be set if bit OSM=1 and bit RST_CCU0 = 0. |
| 20 | RST_CCU0 | Reset source of CCU0<br>0: Reset counter register CN0 to 0 on matching comparison with CM0<br>1: Reset counter register CN0 to 0 on trigger TRIG_[x-1] orTIM_EXT_CAPTURE(x).<br>**NOTES**<br>1. If RST_CCU0=1 and UPEN_CTRLx=1 are set, TRIG_[x-1] or TIM_EXT_CAPTURE(x) triggers also the update of work register (CM0, CM1 and CLK_SRC).<br>2. This bit is only applicable in SOMP mode.<br>3. This bit should only be set if bit OSM=0 (i.e. in continuous mode) |
| 19 to 17 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 16 | WR_REQ | CPU Write request bit for late compare register update.<br>**NOTE**<br>This bit is only applicable in SOMC and SOMB mode. |
| 15 | Reserved | These bits are always read as 0. When written, write the initial value. |

**Table 25.117 GTM0ATOMixCTRL Register Contents (3/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 14 to 12 | CLK_SRC_SR | Actual CMU clock source (SOMS)/ shadow register for CMU clock source (SOMP).<br>000: CMU_CLK0 selected<br>001: CMU_CLK1 selected<br>010: CMU_CLK2 selected<br>011: CMU_CLK3 selected<br>100: CMU_CLK4 selected<br>101: CMU_CLK5 selected<br>110: CMU_CLK6 selected<br>111: CMU_CLK7 selected<br>**NOTES**<br>1. After (channel) reset the selected CLK_SRC value is the SYS_CLK (input of Global Clock Divider). To use in SOMP mode one of the CMU_CLKx, it is required to perform a forced update of CLK_SRC with the value of CLK_SRC_SR value before/with enabling the channel.<br>2. This register is a shadow register for the CMU_CLKx select. Thus, if the CMU_CLK source for PWM generation should be changed during operation, the old CMU_CLK has to operate until the update of the ATOM channels internal CLK_SRC register by the CLK_SRC_SR content is done either by an end of a period or a FORCE_UPDATE. |
| 11 | SL | Initial signal level.<br>0: Low signal level<br>1: High signal level<br>**NOTES**<br>1. Reset value depends on the hardware configuration chosen by silicon vendor.<br>2. If the output is disabled, the output ATOM_OUT[x] is set to inverse SL independent of the ATOM channel mode.<br>3. In SOMP, SOMI, SOMS mode, if the channel is disabled, the internal register SOUR inside ATOM sub unit SOU is set to inverse value of SL. By enabling the channel the register SOUR is not changed. Thus, if the output is enabled afterwards, the output ATOM_OUT[x] is the inverse value of SL.<br>4. In SOMC mode, if the channel is disabled, the internal register SOUR inside ATOM sub unit SOU is set to value of SL. By enabling the channel the register SOUR is not changed. Thus, if the output is enabled and the channel is disabled, the output ATOM_OUT[x] is the value of SL.<br>5. In SOMS mode, this bit is only applicable when the channel and its output are disabled. |
| 10 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 9 | CMP_CTRL | CCUx compare strategy select.<br>0: Greater/equal compare against TBU time base values (TBU_TSx ≥ CMx)<br>1: Less/equal compare against TBU time base values (TBU_TSx ≤ CMx)<br>**NOTE**<br>This bit is only applicable in SOMC mode. |
| 8 to 4 | ACB | ATOM Mode control bits.<br>**NOTES**<br>1. These bits have different meaning in the different ATOM channel modes. See the mode description **Section 25.11.3, ATOM Channel modes** . |
| 3 | ARU_EN | ARU Input stream enable.<br>0: ARU Input stream disabled<br>1: ARU Input stream enabled |

**Table 25.117 GTM0ATOMixCTRL Register Contents (4/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | TB12_SEL | Select time base value TBU_TS1 or TBU_TS2.<br>0: TBU_TS1 selected for comparison<br>1: TBU_TS2 selected for comparison<br>**NOTE**<br>This bit is only applicable in SOMC mode. |
| 1, 0 | MODE | ATOM channel mode select.<br>00: ATOM Signal Output Mode Immediate (SOMI)<br>01: ATOM Signal Output Mode Compare (SOMC)<br>10: ATOM Signal Output Mode PWM (SOMP)<br>11: ATOM Signal Output Mode Serial (SOMS) |

### (2) GTM0ATOMixSTAT (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0ATOM00STAT: <GTM_base> + $0D01C_H$,
GTM0ATOM01STAT: <GTM_base> + $0D09C_H$,
GTM0ATOM02STAT: <GTM_base> + $0D11C_H$,
GTM0ATOM03STAT: <GTM_base> + $0D19C_H$,
GTM0ATOM04STAT: <GTM_base> + $0D21C_H$,
GTM0ATOM05STAT: <GTM_base> + $0D29C_H$,
GTM0ATOM06STAT: <GTM_base> + $0D31C_H$,
GTM0ATOM07STAT: <GTM_base> + $0D39C_H$,
GTM0ATOM10STAT: <GTM_base> + $0D81C_H$
GTM0ATOM11STAT: <GTM_base> + $0D89C_H$
GTM0ATOM12STAT: <GTM_base> + $0D91C_H$
GTM0ATOM13STAT: <GTM_base> + $0D99C_H$
GTM0ATOM14STAT: <GTM_base> + $0DA1C_H$
GTM0ATOM15STAT: <GTM_base> + $0DA9C_H$
GTM0ATOM16STAT: <GTM_base> + $0DB1C_H$
GTM0ATOM17STAT: <GTM_base> + $0DB9C_H$
GTM0ATOM20STAT: <GTM_base> + $0E01C_H$
GTM0ATOM21STAT: <GTM_base> + $0E09C_H$
GTM0ATOM22STAT: <GTM_base> + $0E11C_H$
GTM0ATOM23STAT: <GTM_base> + $0E19C_H$
GTM0ATOM24STAT: <GTM_base> + $0E21C_H$

**Value after reset:** $0000\ 000X_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ACBO | | | | | — | WRF | DV | ACBI | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.118 GTM0ATOMixSTAT Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 28 to 24 | ACBO | ATOM Internal status bits.<br>ACBO[3] = 1: CCU0 Compare match occurred<br>ACBO[4] = 1: CCU1 Compare match occurred<br>**NOTES**<br>1. These bits are only set in SOMC mode.<br>2. ACBO is reset to 0b00000 on an update of register CM0 or CM1 (via ARU or CPU)<br>3. In SOMC mode these bits are sent as ARU control bits 52 to 48. |
| 23 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 22 | WRF | Write request of CPU failed for late update.<br>0: Late update was successful, CCUx units wait for comparison.<br>1: Late update failed.<br>The bit WRF can be reset by writing a 1 to it.<br>**NOTE**<br>This bit is only applicable in SOMC and SOMB mode. |

**Table 25.118 GTM0ATOMixSTAT Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 21 | DV | Valid ARU Data stored in compare registers.<br>0: No valid data stored in register CM0 and/or CM1, no comparison is activated.<br>1: Valid data stored in CM0 and/or CM1, comparison activated.<br>**NOTE**<br>This bit is only applicable in SOMC and SOMB mode. The CPU can determine the status of the ARU transfers with this bit. After the compare event occurred, the bit is reset by hardware. |
| 20 to 16 | ACBI | ATOM Mode control bits.<br>**NOTES**<br>1. For ATOM SOMI, SOMC, SOMP and SOMS mode this register serves as a mirror for the five ARU control bits received through the ARU interface. The bits are valid, when the DV bit is set.<br>2. For SOMB mode this bit field serves as the work register of the compare strategy. It can be updated with the value of bit field ACB of register GTM0ATOMixCTRL or the value of internal shadow register ACB_SR. |
| 15 to 1 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 0 | OL | Actual output signal level of ATOM_CHx_OUT.<br>0: Actual output signal level is low<br>1: Actual output signal level is high<br>**NOTE**<br>Reset value is the inverted value of bit SL which depends on the hardware configuration chosen by silicon vendor. |

### 25.11.6.9 GTM0ATOMixRDADDR (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM00RDADDR: <GTM_base> + $0D000_H$, GTM0ATOM10RDADDR: <GTM_base> + $0D800_H$
GTM0ATOM01RDADDR: <GTM_base> + $0D080_H$, GTM0ATOM11RDADDR: <GTM_base> + $0D880_H$
GTM0ATOM02RDADDR: <GTM_base> + $0D100_H$, GTM0ATOM12RDADDR: <GTM_base> + $0D900_H$
GTM0ATOM03RDADDR: <GTM_base> + $0D180_H$, GTM0ATOM13RDADDR: <GTM_base> + $0D980_H$
GTM0ATOM04RDADDR: <GTM_base> + $0D200_H$, GTM0ATOM14RDADDR: <GTM_base> + $0DA00_H$
GTM0ATOM05RDADDR: <GTM_base> + $0D280_H$, GTM0ATOM15RDADDR: <GTM_base> + $0DA80_H$
GTM0ATOM06RDADDR: <GTM_base> + $0D300_H$, GTM0ATOM16RDADDR: <GTM_base> + $0DB00_H$
GTM0ATOM07RDADDR: <GTM_base> + $0D380_H$, GTM0ATOM17RDADDR: <GTM_base> + $0DB80_H$
GTM0ATOM20RDADDR: <GTM_base> + $0E000_H$
GTM0ATOM21RDADDR: <GTM_base> + $0E080_H$
GTM0ATOM22RDADDR: <GTM_base> + $0E100_H$
GTM0ATOM23RDADDR: <GTM_base> + $0E180_H$
GTM0ATOM24RDADDR: <GTM_base> + $0E200_H$

**Value after reset:** $01FE\ 01FE_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | RDADDR1 | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | RDADDR0 | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.119 GTM0ATOMixRDADDR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 25 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 24 to 16 | RDADDR1 | ARU Read address 1.<br>**NOTES**<br>1. The ATOM channel switches to this read address, when requested in the ARU control bits 52 to 48 with the pattern “111--”. The channel switches back to the RDADDR0 after one ARU data package was received on RDADDR1.<br>2. This read address is only applicable in SOMC mode.<br>3. This bit field is only writeable if channel is disabled. |
| 15 to 9 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 8 to 0 | RDADDR0 | ARU Read address 0.<br>**NOTES**<br>1. This read address is used by the ATOM channel to receive data from ARU immediately after the channel and ARU access is enabled (see **Section (1), GTM0ATOMixCTRL (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)** for details).<br>2. This bit field is only writeable if channel is disabled. |

### 25.11.6.10 GTM0ATOMixCN0 (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0ATOM00CN0: <GTM_base> + $0D018_H$,
GTM0ATOM01CN0: <GTM_base> + $0D098_H$,
GTM0ATOM02CN0: <GTM_base> + $0D118_H$,
GTM0ATOM03CN0: <GTM_base> + $0D198_H$,
GTM0ATOM04CN0: <GTM_base> + $0D218_H$,
GTM0ATOM05CN0: <GTM_base> + $0D298_H$,
GTM0ATOM06CN0: <GTM_base> + $0D318_H$,
GTM0ATOM07CN0: <GTM_base> + $0D398_H$,
GTM0ATOM20CN0: <GTM_base> + $0E018_H$
GTM0ATOM21CN0: <GTM_base> + $0E098_H$
GTM0ATOM22CN0: <GTM_base> + $0E118_H$
GTM0ATOM23CN0: <GTM_base> + $0E198_H$
GTM0ATOM24CN0: <GTM_base> + $0E218_H$
GTM0ATOM10CN0: <GTM_base> + $0D818_H$
GTM0ATOM11CN0: <GTM_base> + $0D898_H$
GTM0ATOM12CN0: <GTM_base> + $0D918_H$
GTM0ATOM13CN0: <GTM_base> + $0D998_H$
GTM0ATOM14CN0: <GTM_base> + $0DA18_H$
GTM0ATOM15CN0: <GTM_base> + $0DA98_H$
GTM0ATOM16CN0: <GTM_base> + $0DB18_H$
GTM0ATOM17CN0: <GTM_base> + $0DB98_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CN0 | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CN0 | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.120 GTM0ATOMixCN0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | CN0 | ATOM CCU0 counter register. |

### 25.11.6.11 GTM0ATOMixCM0 (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0ATOM00CM0: <GTM_base> + $0D010_H$,
GTM0ATOM01CM0: <GTM_base> + $0D090_H$,
GTM0ATOM02CM0: <GTM_base> + $0D110_H$,
GTM0ATOM03CM0: <GTM_base> + $0D190_H$,
GTM0ATOM04CM0: <GTM_base> + $0D210_H$,
GTM0ATOM05CM0: <GTM_base> + $0D290_H$,
GTM0ATOM06CM0: <GTM_base> + $0D310_H$,
GTM0ATOM07CM0: <GTM_base> + $0D390_H$,
GTM0ATOM20CM0: <GTM_base> + $0E010_H$
GTM0ATOM21CM0: <GTM_base> + $0E090_H$
GTM0ATOM22CM0: <GTM_base> + $0E110_H$
GTM0ATOM23CM0: <GTM_base> + $0E190_H$
GTM0ATOM24CM0: <GTM_base> + $0E210_H$
GTM0ATOM10CM0: <GTM_base> + $0D810_H$
GTM0ATOM11CM0: <GTM_base> + $0D890_H$
GTM0ATOM12CM0: <GTM_base> + $0D910_H$
GTM0ATOM13CM0: <GTM_base> + $0D990_H$
GTM0ATOM14CM0: <GTM_base> + $0DA10_H$
GTM0ATOM15CM0: <GTM_base> + $0DA90_H$
GTM0ATOM16CM0: <GTM_base> + $0DB10_H$
GTM0ATOM17CM0: <GTM_base> + $0DB90_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CM0 | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CM0 | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.121 GTM0ATOMixCM0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | CM0 | ATOM CCU0 compare register. |

**NOTE**

This register is write protected in SOMC mode and returns AEI_STATUS = ‘10’ on write access, when in serve last compare strategy the first match of CCU0 occurred.

### 25.11.6.12 GTM0ATOMixSR0 (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0ATOM00SR0: <GTM_base> + $0D008_H$,
GTM0ATOM01SR0: <GTM_base> + $0D088_H$,
GTM0ATOM02SR0: <GTM_base> + $0D108_H$,
GTM0ATOM03SR0: <GTM_base> + $0D188_H$,
GTM0ATOM04SR0: <GTM_base> + $0D208_H$,
GTM0ATOM05SR0: <GTM_base> + $0D288_H$,
GTM0ATOM06SR0: <GTM_base> + $0D308_H$,
GTM0ATOM07SR0: <GTM_base> + $0D388_H$,
GTM0ATOM10SR0: <GTM_base> + $0D808_H$
GTM0ATOM11SR0: <GTM_base> + $0D888_H$
GTM0ATOM12SR0: <GTM_base> + $0D908_H$
GTM0ATOM13SR0: <GTM_base> + $0D988_H$
GTM0ATOM14SR0: <GTM_base> + $0DA08_H$
GTM0ATOM15SR0: <GTM_base> + $0DA88_H$
GTM0ATOM16SR0: <GTM_base> + $0DB08_H$
GTM0ATOM17SR0: <GTM_base> + $0DB88_H$
GTM0ATOM20SR0: <GTM_base> + $0E008_H$
GTM0ATOM21SR0: <GTM_base> + $0E088_H$
GTM0ATOM22SR0: <GTM_base> + $0E108_H$
GTM0ATOM23SR0: <GTM_base> + $0E188_H$
GTM0ATOM24SR0: <GTM_base> + $0E208_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | SR0 | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR0 | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.122 GTM0ATOMixSR0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | SR0 | ATOM channel x shadow register SR0. |

**NOTE**

The SR0 register is used as shadow register for CM0 in SOMP and SOMS modes and is used as capture register for time base TBU_TS0 in SOMC mode.

### 25.11.6.13GTM0ATOMixCM1 (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0ATOM00CM1: <GTM_base> + $0D014_H$,
GTM0ATOM01CM1: <GTM_base> + $0D094_H$,
GTM0ATOM02CM1: <GTM_base> + $0D114_H$,
GTM0ATOM03CM1: <GTM_base> + $0D194_H$,
GTM0ATOM04CM1: <GTM_base> + $0D214_H$,
GTM0ATOM05CM1: <GTM_base> + $0D294_H$,
GTM0ATOM06CM1: <GTM_base> + $0D314_H$,
GTM0ATOM07CM1: <GTM_base> + $0D394_H$,
GTM0ATOM20CM1: <GTM_base> + $0E014_H$
GTM0ATOM21CM1: <GTM_base> + $0E094_H$
GTM0ATOM22CM1: <GTM_base> + $0E114_H$
GTM0ATOM23CM1: <GTM_base> + $0E194_H$
GTM0ATOM24CM1: <GTM_base> + $0E214_H$
GTM0ATOM10CM1: <GTM_base> + $0D814_H$
GTM0ATOM11CM1: <GTM_base> + $0D894_H$
GTM0ATOM12CM1: <GTM_base> + $0D914_H$
GTM0ATOM13CM1: <GTM_base> + $0D994_H$
GTM0ATOM14CM1: <GTM_base> + $0DA14_H$
GTM0ATOM15CM1: <GTM_base> + $0DA94_H$
GTM0ATOM16CM1: <GTM_base> + $0DB14_H$
GTM0ATOM17CM1: <GTM_base> + $0DB94_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CM1 | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CM1 | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.123 GTM0ATOMixCM1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | CM1 | ATOM CCU1 compare register. |

**NOTE**

This register is write protected in SOMC mode and returns AEI_STATUS = ‘10’ on write access, when in serve last compare strategy the first match of CCU0 occurred.

### 25.11.6.14 GTM0ATOMixSR1 (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0ATOM00SR1: <GTM_base> + $0D00C_H$,
GTM0ATOM01SR1: <GTM_base> + $0D08C_H$,
GTM0ATOM02SR1: <GTM_base> + $0D10C_H$,
GTM0ATOM03SR1: <GTM_base> + $0D18C_H$,
GTM0ATOM04SR1: <GTM_base> + $0D20C_H$,
GTM0ATOM05SR1: <GTM_base> + $0D28C_H$,
GTM0ATOM06SR1: <GTM_base> + $0D30C_H$,
GTM0ATOM07SR1: <GTM_base> + $0D38C_H$,
GTM0ATOM10SR1: <GTM_base> + $0D80C_H$
GTM0ATOM11SR1: <GTM_base> + $0D88C_H$
GTM0ATOM12SR1: <GTM_base> + $0D90C_H$
GTM0ATOM13SR1: <GTM_base> + $0D98C_H$
GTM0ATOM14SR1: <GTM_base> + $0DA0C_H$
GTM0ATOM15SR1: <GTM_base> + $0DA8C_H$
GTM0ATOM16SR1: <GTM_base> + $0DB0C_H$
GTM0ATOM17SR1: <GTM_base> + $0DB8C_H$
GTM0ATOM20SR1: <GTM_base> + $0E00C_H$
GTM0ATOM21SR1: <GTM_base> + $0E08C_H$
GTM0ATOM22SR1: <GTM_base> + $0E10C_H$
GTM0ATOM23SR1: <GTM_base> + $0E18C_H$
GTM0ATOM24SR1: <GTM_base> + $0E20C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | SR1 | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR1 | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.124 GTM0ATOMixSR1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | SR1 | ATOM channel x shadow register SR0. |

**NOTE**

The SR1 register is used as shadow register for CM1 in SOMP and SOMS modes and is used as capture register for time base TBU_TS1 or TBU_TS2 (when selected in GTM0ATOMixCTRL register) in SOMC mode.

### 25.11.6.15 GTM0ATOMixIRQNOTIFY (x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM00IRQNOTIFY: <GTM_base> + $0D020_H$,GTM0ATOM10IRQNOTIFY: <GTM_base> + $0D820_H$
GTM0ATOM01IRQNOTIFY: <GTM_base> + $0D0A0_H$,GTM0ATOM11IRQNOTIFY: <GTM_base> + $0D8A0_H$
GTM0ATOM02IRQNOTIFY: <GTM_base> + $0D120_H$,GTM0ATOM12IRQNOTIFY: <GTM_base> + $0D920_H$
GTM0ATOM03IRQNOTIFY: <GTM_base> + $0D1A0_H$,GTM0ATOM13IRQNOTIFY: <GTM_base> + $0D9A0_H$
GTM0ATOM04IRQNOTIFY: <GTM_base> + $0D220_H$,GTM0ATOM14IRQNOTIFY: <GTM_base> + $0DA20_H$
GTM0ATOM05IRQNOTIFY: <GTM_base> + $0D2A0_H$,GTM0ATOM15IRQNOTIFY: <GTM_base> + $0DAA0_H$
GTM0ATOM06IRQNOTIFY: <GTM_base> + $0D320_H$,GTM0ATOM16IRQNOTIFY: <GTM_base> + $0DB20_H$
GTM0ATOM07IRQNOTIFY: <GTM_base> + $0D3A0_H$,GTM0ATOM17IRQNOTIFY: <GTM_base> + $0DBA0_H$
GTM0ATOM20IRQNOTIFY: <GTM_base> + $0E020_H$
GTM0ATOM21IRQNOTIFY: <GTM_base> + $0E0A0_H$
GTM0ATOM22IRQNOTIFY: <GTM_base> + $0E120_H$
GTM0ATOM23IRQNOTIFY: <GTM_base> + $0E1A0_H$
GTM0ATOM24IRQNOTIFY: <GTM_base> + $0E220_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CCU1TC | CCU0TC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 25.125 GTM0ATOMixIRQNOTIFY Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1 | CCU1TC | CCU1 Trigger condition interrupt for channel x.<br>See bit 0. |
| 0 | CCU0TC | CCU0 Trigger condition interrupt for channel x.<br>0: No interrupt occurred.<br>1: CCU0 Trigger condition interrupt was raised by ATOM channel x.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

### 25.11.6.16 GTM0ATOMixIRQEN (x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0ATOM00IRQEN: <GTM_base> + $0D024_H$,
GTM0ATOM01IRQEN: <GTM_base> + $0D0A4_H$,
GTM0ATOM02IRQEN: <GTM_base> + $0D124_H$,
GTM0ATOM03IRQEN: <GTM_base> + $0D1A4_H$,
GTM0ATOM04IRQEN: <GTM_base> + $0D224_H$,
GTM0ATOM05IRQEN: <GTM_base> + $0D2A4_H$,
GTM0ATOM06IRQEN: <GTM_base> + $0D324_H$,
GTM0ATOM07IRQEN: <GTM_base> + $0D3A4_H$,
GTM0ATOM10IRQEN: <GTM_base> + $0D824_H$
GTM0ATOM11IRQEN: <GTM_base> + $0D8A4_H$
GTM0ATOM12IRQEN: <GTM_base> + $0D924_H$
GTM0ATOM13IRQEN: <GTM_base> + $0D9A4_H$
GTM0ATOM14IRQEN: <GTM_base> + $0DA24_H$
GTM0ATOM15IRQEN: <GTM_base> + $0DAA4_H$
GTM0ATOM16IRQEN: <GTM_base> + $0DB24_H$
GTM0ATOM17IRQEN: <GTM_base> + $0DBA4_H$
GTM0ATOM20IRQEN: <GTM_base> + $0E024_H$
GTM0ATOM21IRQEN: <GTM_base> + $0E0A4_H$
GTM0ATOM22IRQEN: <GTM_base> + $0E124_H$
GTM0ATOM23IRQEN: <GTM_base> + $0E1A4_H$
GTM0ATOM24IRQEN: <GTM_base> + $0E224_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CCU1TC_IRQ_EN | CCU0TC_IRQ_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 25.126 GTM0ATOMixIRQEN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1 | CCU1TC_IRQ_EN | ATOM_CCU1TC_IRQ interrupt enable.<br>See bit 0. |
| 0 | CCU0TC_IRQ_EN | ATOM_CCU0TC_IRQ interrupt enable.<br>0: Disable interrupt, interrupt is not visible outside GTM-IP.<br>1: Enable interrupt, interrupt is visible outside GTM-IP. |

## 25.11.6.17GTM0ATOMixIRQFORCINT (x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM00IRQFORCINT: <GTM_base> + $0D028_H$,GTM0ATOM10IRQFORCINT: <GTM_base> + $0D828_H$
GTM0ATOM01IRQFORCINT: <GTM_base> + $0D0A8_H$,GTM0ATOM11IRQFORCINT: <GTM_base> + $0D8A8_H$
GTM0ATOM02IRQFORCINT: <GTM_base> + $0D128_H$,GTM0ATOM12IRQFORCINT: <GTM_base> + $0D928_H$
GTM0ATOM03IRQFORCINT: <GTM_base> + $0D1A8_H$,GTM0ATOM13IRQFORCINT: <GTM_base> + $0D9A8_H$
GTM0ATOM04IRQFORCINT: <GTM_base> + $0D228_H$,GTM0ATOM14IRQFORCINT: <GTM_base> + $0DA28_H$
GTM0ATOM05IRQFORCINT: <GTM_base> + $0D2A8_H$,GTM0ATOM15IRQFORCINT: <GTM_base> + $0DAA8_H$
GTM0ATOM06IRQFORCINT: <GTM_base> + $0D328_H$,GTM0ATOM16IRQFORCINT: <GTM_base> + $0DB28_H$
GTM0ATOM07IRQFORCINT: <GTM_base> + $0D3A8_H$,GTM0ATOM17IRQFORCINT: <GTM_base> + $0DBA8_H$
GTM0ATOM20IRQFORCINT: <GTM_base> + $0E028_H$
GTM0ATOM21IRQFORCINT: <GTM_base> + $0E0A8_H$
GTM0ATOM22IRQFORCINT: <GTM_base> + $0E128_H$
GTM0ATOM23IRQFORCINT: <GTM_base> + $0E1A8_H$
GTM0ATOM24IRQFORCINT: <GTM_base> + $0E228_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TRG_CCU1TC | TRG_CCU0TC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 25.127 GTM0ATOMixIRQFORCINT (x = 0 to 7) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1 | TRG_CCU1TC | Trigger ATOM_CCU1TC_IRQ interrupt by software<br>0: No interrupt triggering.<br>1: Assert CCU1TC_IRQ interrupt for one clock cycle.<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of register GTM0GTMCTRL. |
| 0 | TRG_CCU0TC | Trigger ATOM_CCU0TC_IRQ interrupt by software.<br>0: No interrupt triggering.<br>1: Assert CCU0TC_IRQ interrupt for one clock cycle.<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of register GTM0GTMCTRL. |

## 25.11.6.18GTM0ATOM00IRQMOD (x = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM00IRQMODE: <GTM_base> + $0D02C_H$,GTM0ATOM10IRQMODE: <GTM_base> + $0D82C_H$
GTM0ATOM01IRQMODE: <GTM_base> + $0D0AC_H$,GTM0ATOM11IRQMODE: <GTM_base> + $0D8AC_H$
GTM0ATOM02IRQMODE: <GTM_base> + $0D12C_H$,GTM0ATOM12IRQMODE: <GTM_base> + $0D92C_H$
GTM0ATOM03IRQMODE: <GTM_base> + $0D1AC_H$,GTM0ATOM13IRQMODE: <GTM_base> + $0D9AC_H$
GTM0ATOM04IRQMODE: <GTM_base> + $0D22C_H$,GTM0ATOM14IRQMODE: <GTM_base> + $0DA2C_H$
GTM0ATOM05IRQMODE: <GTM_base> + $0D2AC_H$,GTM0ATOM15IRQMODE: <GTM_base> + $0DAAC_H$
GTM0ATOM06IRQMODE: <GTM_base> + $0D32C_H$,GTM0ATOM16IRQMODE: <GTM_base> + $0DB2C_H$
GTM0ATOM07IRQMODE: <GTM_base> + $0D3AC_H$,GTM0ATOM17IRQMODE: <GTM_base> + $0DBAC_H$
GTM0ATOM20IRQMODE: <GTM_base> + $0E02C_H$
GTM0ATOM21IRQMODE: <GTM_base> + $0E0AC_H$
GTM0ATOM22IRQMODE: <GTM_base> + $0E12C_H$
GTM0ATOM23IRQMODE: <GTM_base> + $0E1AC_H$
GTM0ATOM24IRQMODE: <GTM_base> + $0E22C_H$

**Value after reset:** $0000\ 000X_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IRQ_MODE | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 25.128 GTM0ATOM00IRQMOD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1, 0 | IRQ_MODE | IRQ mode selection<br>00: Level mode<br>01: Pulse mode<br>10: Pulse-Notify mode<br>11: Single-Pulse mode<br>**NOTE**<br>The interrupt modes are described in **Section 25.6.5, GTM-IP Interrupt Concept** . |

# 25.12 Dead Time Module (DTM)

## 25.12.1 Overview

The following figure gives an overview of the structure of the Dead Time Module (DTM).

### 25.12.1.1 DTM overview

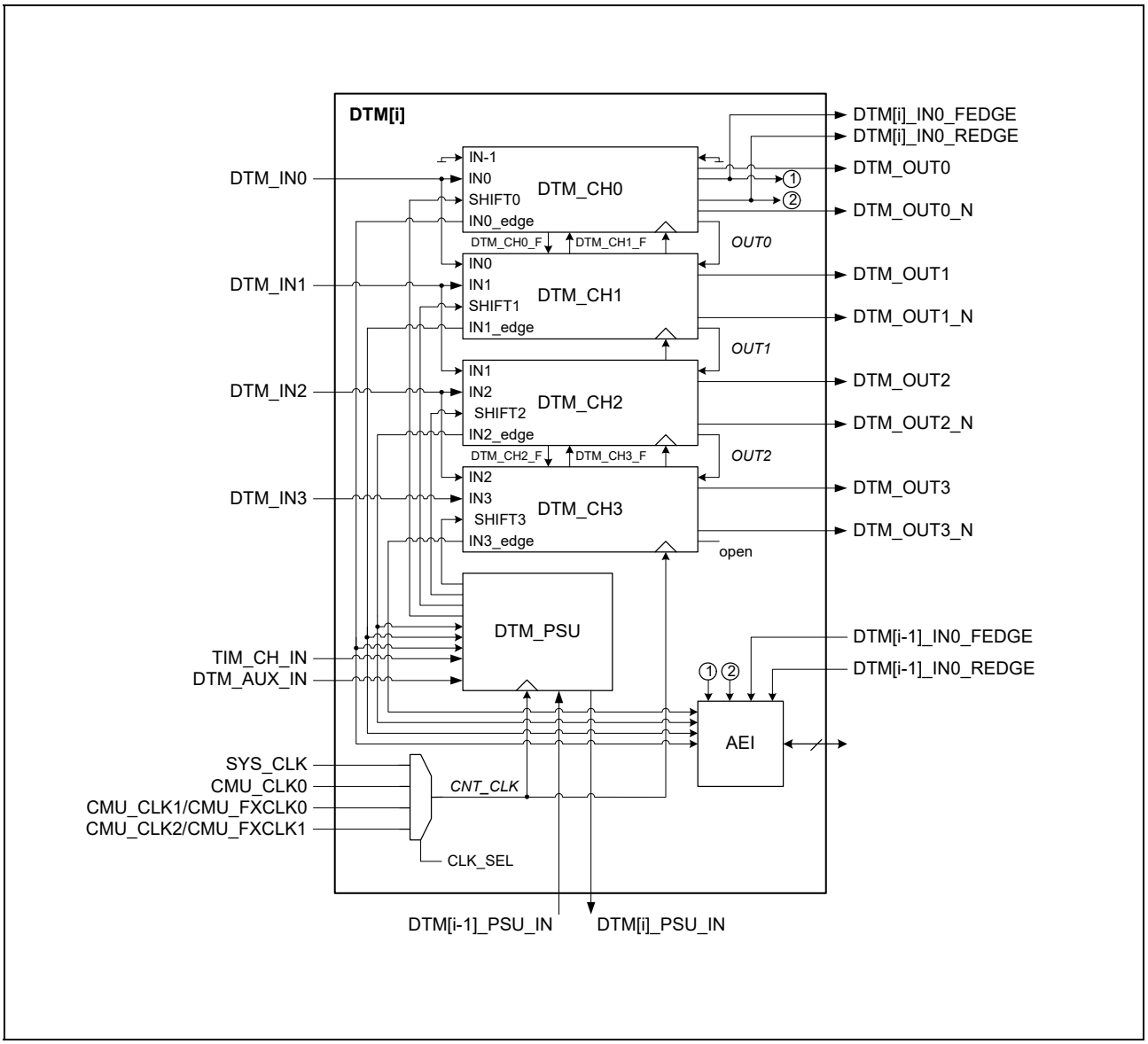



**Figure 25.52 DTM block diagram**

The main function of the DTM is to derive for each input DTM_IN0 to DTM_IN3 the individual inverse signal (DTM[i]_OUT[x]_N) and to apply an edge specific delay between the edge of the original signal and the edge of the derived inverted signal (i.e., the dead time). This function is mainly used for controlling of half bridges.

A second function provided by DTM is to set the outputs of one channel to the value of the preceding channel if requested by a trigger on input TIM_CH_IN or DTM_AUX_IN. This feature allows a phase shift on one PWM signal to the phase of the preceding PWM signal up to the next edge on this channel.

The third function provided by DTM is to (N)AND/(N)OR/X(N)OR combine the input DTM_IN[x] signal of one DTM channel with the signal on input TIM_CH_IN or DTM_AUX_IN (selected inside DTM_PSU and assigned to one of the signals SHIFT[x]) or with the output 1 (signal OUT[x]) of preceding channel.

As a result OUT2 may be the combined signal of DTM_IN0 and TIM_CH_IN or DTM_AUX_IN and the signal DTM_IN1. For OUT3 this chain can be combined again with signal DTM_IN3.

The outputs of each channel may be swapped individually to provide the function of combining signals on each output of a channel.

In general, the DTM instances are placed behind the TOM and the ATOM instances, i.e., the outputs TOM_OUT[x] and ATOM_OUT[x] are each routed to a DTM instance.

Additionally, some TIM instances are also connected to the DTM instances.

These connections between DTM and the modules TIM, TOM and ATOM are depicted in **25.12.1.2**.

Note, for unavailable DTM[i] instances the signal DTM[i-1]_PSU_IN is passed through to DTM[i]_PSU_IN, DTM[i-1]_IN0_FEDGE and DTM[i-1]_IN0_REDGE are passed through to DTM[i]_IN0_FEDGE and DTM[i]_IN0_REDGE.

Note, depending on device configuration, not every DTM instance is available. E.g. a device may only have one DTM connected to the first four channels of ATOM. In this case, the other four channels (4 to 7) are connected directly to GTM outputs.

For detailed information, which DTM instance is available, see corresponding device specific **Table 25.2, Sub-Units and Channels** of this specification [1].

### 25.12.1.2 Connections of TIM, TOM and ATOM to DTM

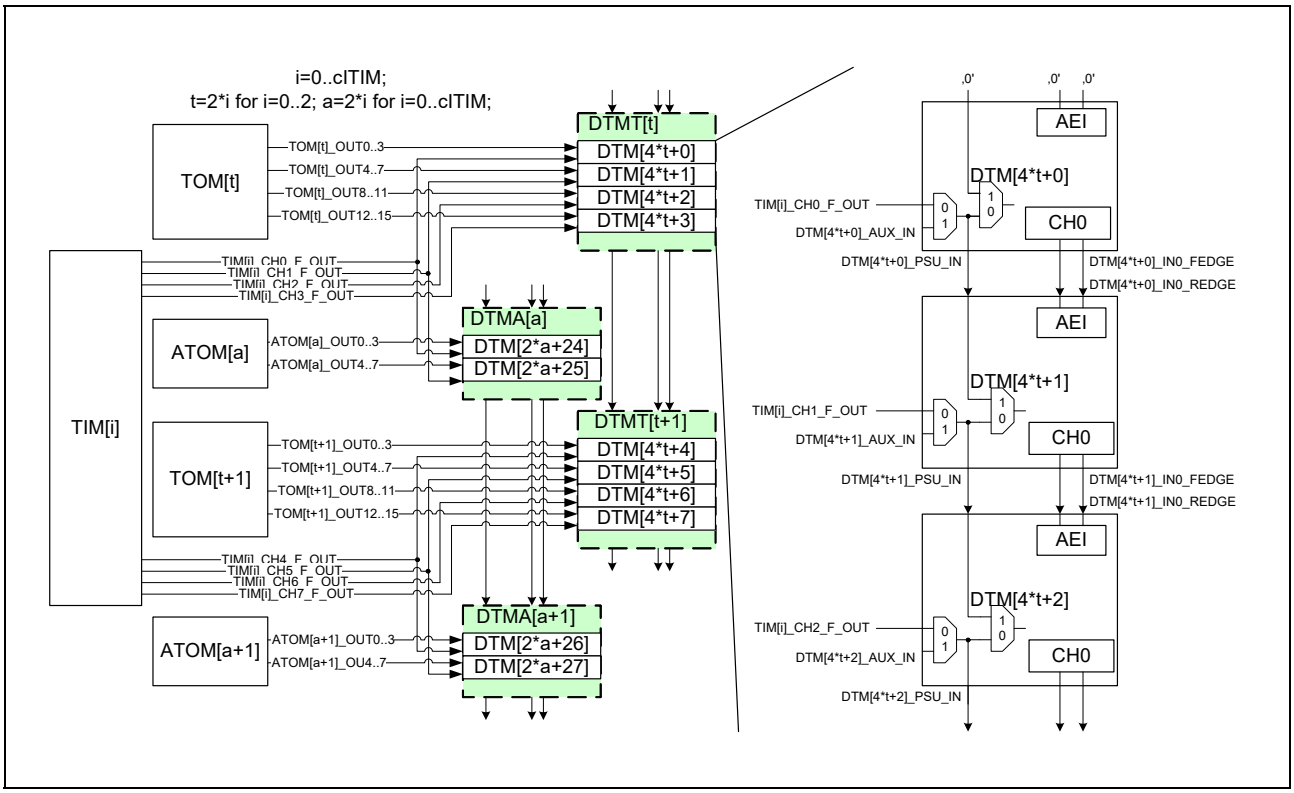



**Figure 25.53 Connections of TIM, TOM and ATOM to DTM**

For each TIM instance index i it can be calculates which instance of DTM and thus also which DTM channel is connected to which TIM channel, TOM channel or ATOM channel.

Four DTM instances behind a TOM instance x are grouped together in a hierarchy called DTMT[x].

Two DTM instances behind an ATOM instance x are grouped together in a hierarchy called DTMA[x].

A DTM instance is only available if there exist at least one TOM or ATOM instance and one TIM instance that can be connected to it.

## 25.12.2 DTM Channel

The following figure depicts the functions of a DTM channel.

### 25.12.2.1 DTM channel 0 overview

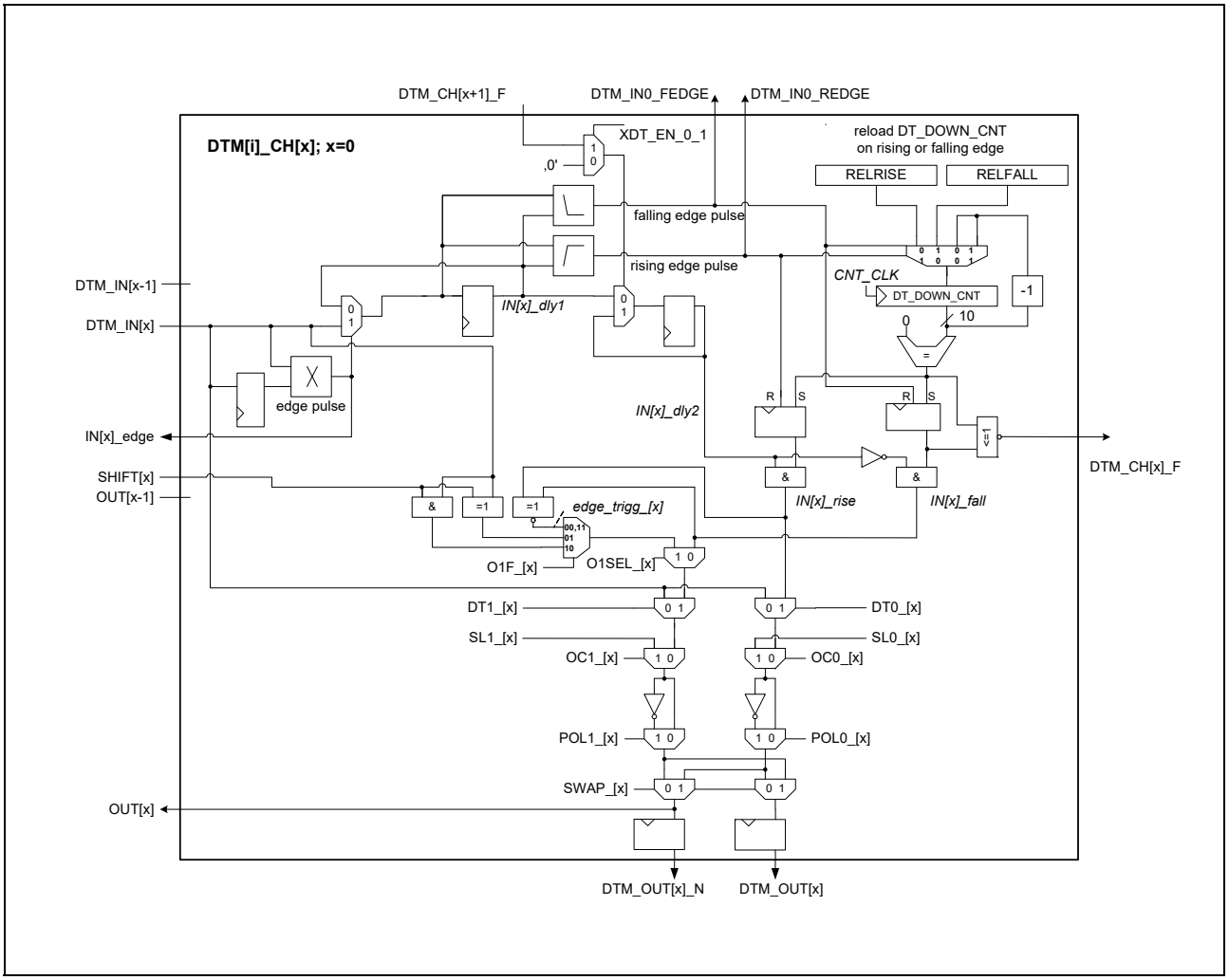



**Figure 25.54 DTM channel 0 block diagram**

### 25.12.2.2 DTM channel 1 to 3 overview

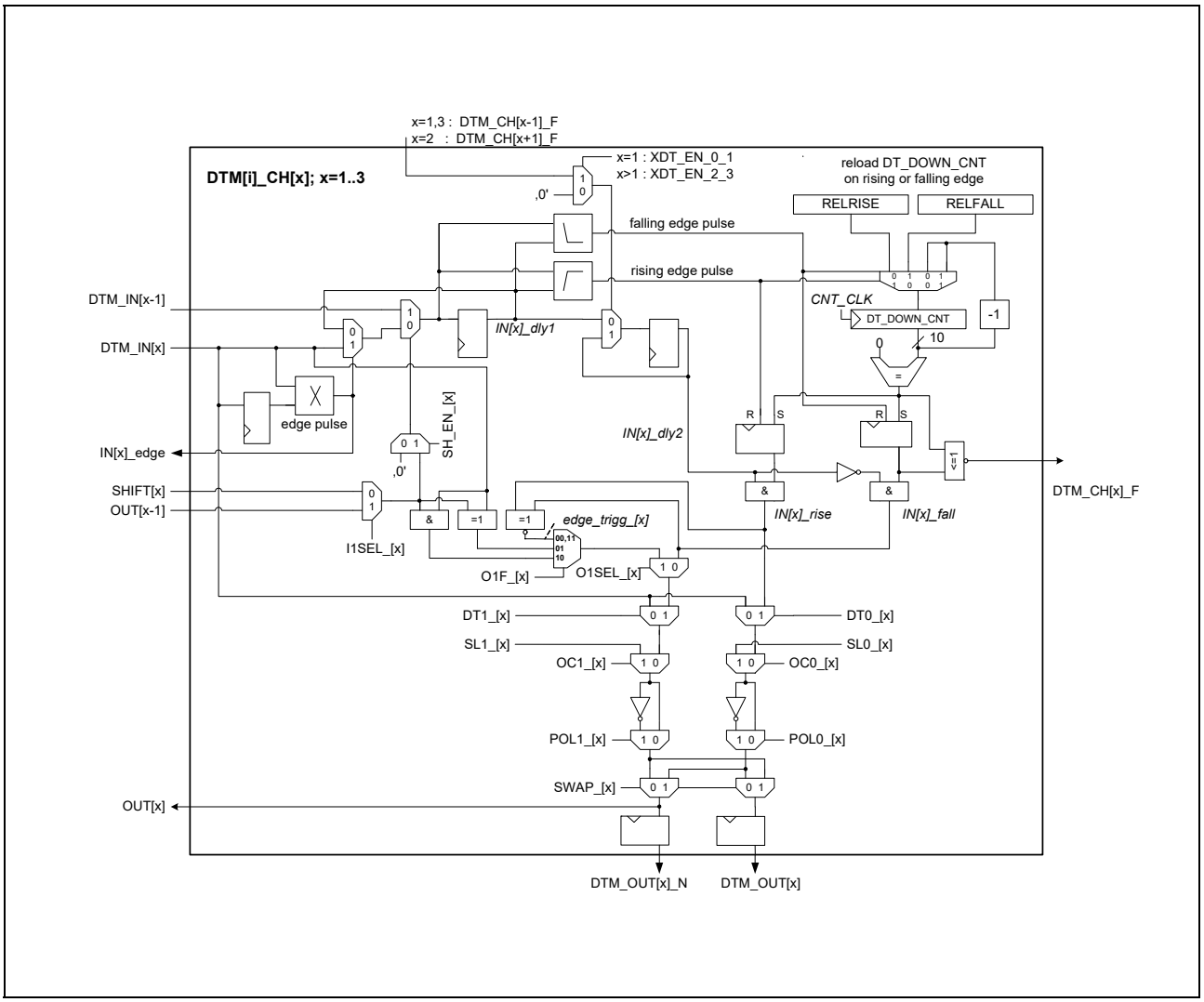



**Figure 25.55 DTM channel 1 to 3 block diagram**

The main feature of each channel is to derive the inverse signal out of the input signal DTM_IN[x], apply an edge dependent delay on the two resulting signal paths and provide these signals at the outputs DTM[i]_OUT[x] and DTM[i]_OUT[x]_N.

There are two possibilities to apply dead time on GTM output signals.

One is to use one DTM channel per TOM/ATOM channel and generate inside the DTM the second inverse signal. This is called the standard dead time generation.

The second way is to generate two signals out of two TOM/ATOM channel and to apply inside the DTM only the dead time by using two cross linked DTM channel. This is called the cross dead time generation.

### 25.12.2.3 Standard dead time generation

Standard dead time generation means that per DTM channel out of one input signal the inverse output signal is generated additionally and on both output signals the dead time between their edges is applied.

The dead time can be configured for each edge individually. The bit field RELRISE in register GTM0DTMixDTV contains the reload value for the counter and defines the delay for rising edges in multiples of selected clock ticks.

The bit field RELFALL in register GTM0DTMixDTV contains the reload value for the counter and defines the delay for falling edges in multiples of selected clock ticks.

The counter is reloaded with the value of RELRISE on a rising edge and reloaded with the value of RELFALL on a falling edge on input DTM_IN[x] (or DTM_IN[x-1] in case of shift enable SH_EN_[x]).

On a reload of the counter the FlipFlop following the counter output comparator is reset and stays reset until the counter has reached 0.

After reload, the counter DT_DOWN_CNT counts down until it reaches 0 and stops at 0.

The signal flow for function of standard dead time signal generation is depicted in following figure

### 25.12.2.4 Wave signals for function of dead time generation

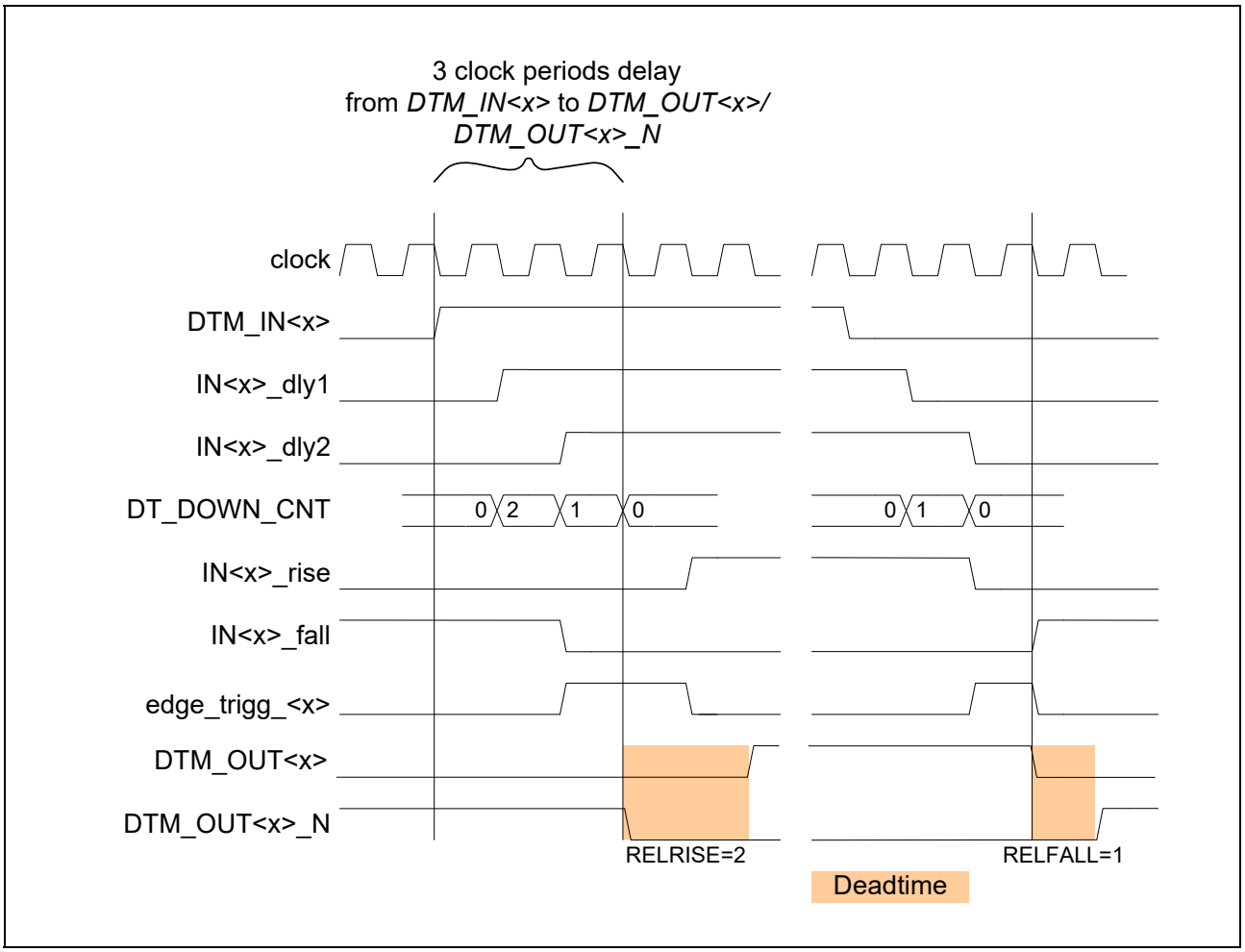



Figure 25.56 Wave signals for function of dead time generation

**NOTES**

1. The delay from the input signal DTM_IN[x] to the output signals DTM[i]_OUT[x] and DTM[i]_OUT[x]_N is three system clock periods by disabled feed through (see DT0/1_[x] in GTM0DTMiCHCTRL2).
2. The delay from the input signal DTM_IN[x] to the output signals DTM[i]_OUT[x] and DTM[i]_OUT[x]_N is one system clock periods by enabled feed through (see DT0/1_[x] in GTM0DTMiCHCTRL2).
3. The reset level of the output signals DTM[i]_OUT[x] connected from ATOM module depends on the hardware configuration value atom_out_reset_level_c chosen by silicon vendor.
4. The reset level of the output signals DTM[i]_OUT[x]_N connected from ATOM module depends on the inverted hardware configuration value atom_out_reset_level_c chosen by silicon vendor.

#### 25.12.2.5 Cross channel dead time

A second way to apply a dead time value on two output signals is the cross channel dead time.

In opposite to the dead time described in **25.12.2.3** the cross channel dead time mode does not generate out of one signal the corresponding inverse signal but tries to apply the dead time on the input signals of two neighbored DTM channel.

To do this, two neighbored DTM input signals (on DTM channel 0 and 1 or on DTM channel 2 and 3) are cross linked together in the way that a falling edge on one channel leads to a hold phase of current signal value on the cross linked channel.

This behavior is reached by the following:

A falling edge on e.g. channel 0 reloads the DT_DOWN_CNT with the value of RELFALL. While this counter is counting down, the output signal of the cross linked channel 1 keeps its value. If the counter DT_DOWN_CNT has reached 0 again, the channel 1 output is released and can follow the value on its input.

The timing of the cross channel dead time is depicted in the following figure:

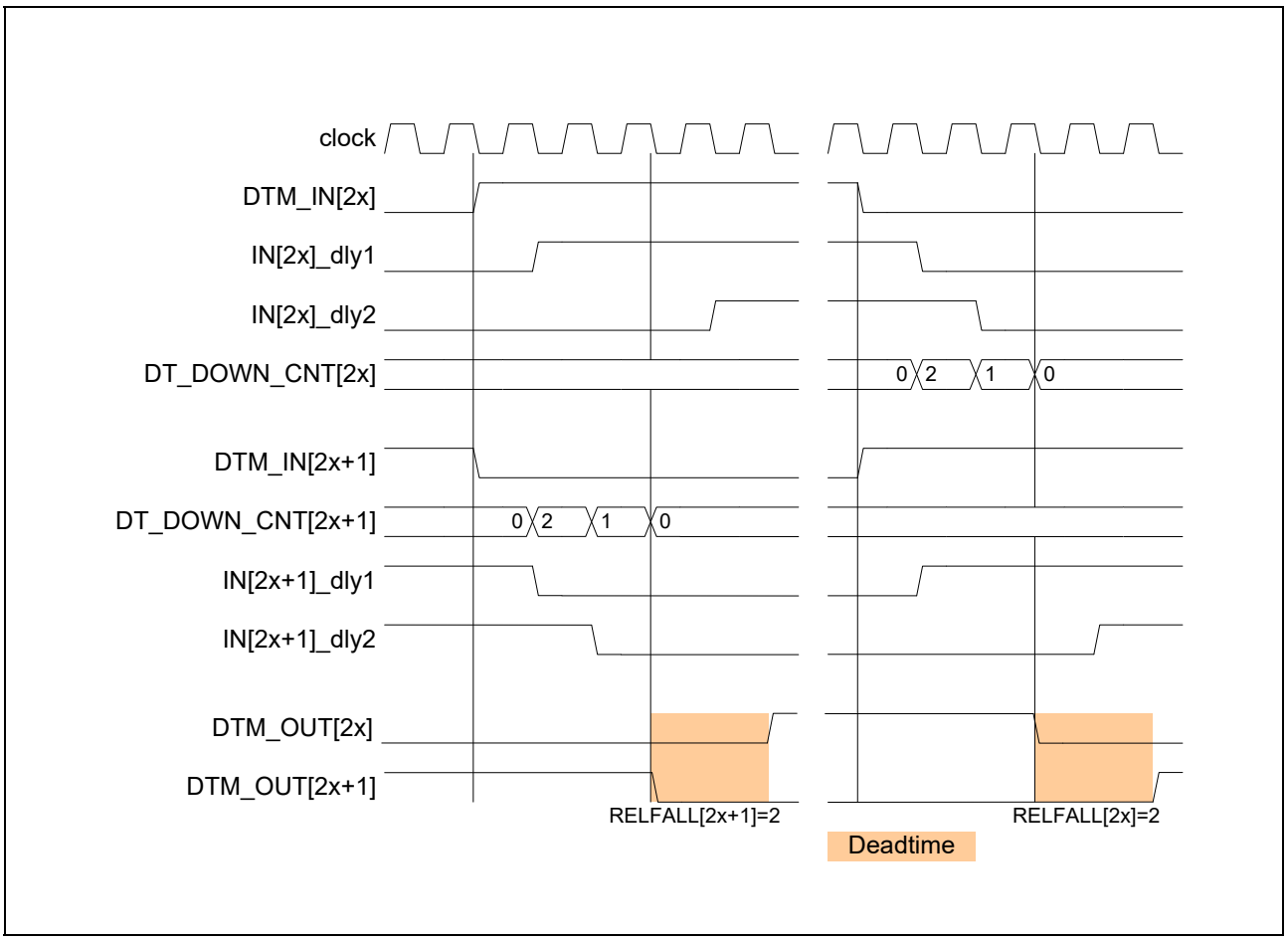



**Figure 25.57 Cross channel dead time timing diagram**

## 25.12.3 Phase Shift Control Unit

The phase shift unit (DTM_PSU) is depicted in the following figure. It supports the second major function of the DTM module to allow phase shifting of PWM signal on one of the channels.

### 25.12.3.1 Phase Shift Unit overview

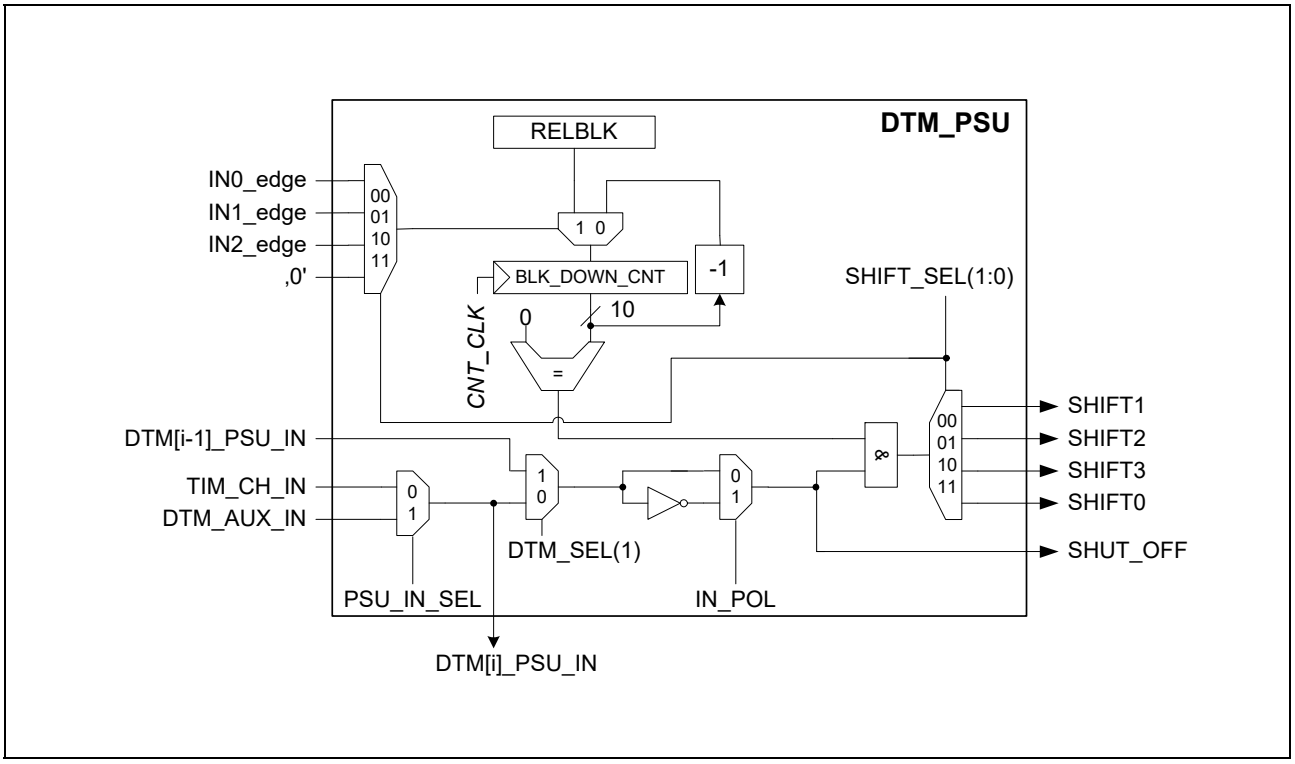



**Figure 25.58 Phase Shift Unit overview**

This submodule provides an additional counter BLK_DOWN_CNT and reload register RELBLK (bit field of register GTM0DTMiPSCTRL). The counter is reloaded on an edge detected on one of the selected signals IN0_edge to IN2_edge (selected by bit field SHIFT_SEL in register GTM0DTMiPSCTRL). Then, the counter counts down until it reaches 0. While the counter is counting down, it blocks the trigger (i.e. the selected one of the signals SHIFT[x]) of one of the channels by one of the input signals TIM_CH_IN or DTM_AUX_IN.

If the counter BLK_DOWN_CNT is not counting, a pulse on the input TIM_CH_IN or DTM_AUX_IN is forwarded to one of the selected DTM_PSU outputs SHIFT[x]. This signal triggers in the selected channel (if SH_EN_x=1) the update of the first Flip-Flop on channel x (i.e. representing IN[x]_DLY ) to the input value DTM_IN[x-1] of the preceding channel. If this update leads to an edge, the succeeding part of DTM channel derives the inverse signal and applies the corresponding dead time (i.e. the edge delay) to the output signals of the channel.

**NOTE**

For channel x=0 input signals DTM_IN[x-1] and OUT[x-1] are unused and I1SEL_[x] and SH_EN_[x] are defined as 0.

The following figure shows an example of phase shifting on channel 1.

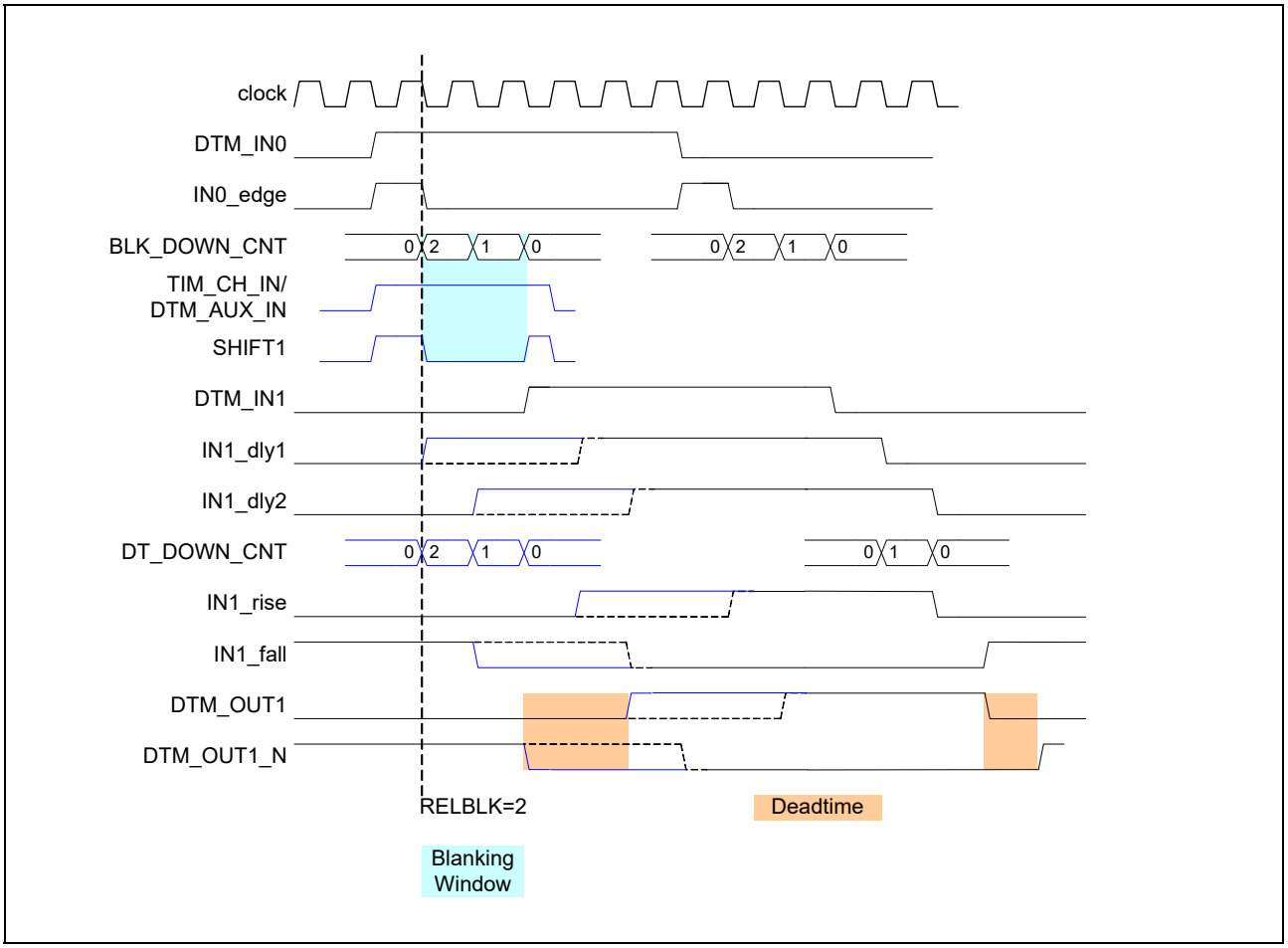



**Figure 25.59 Example wave of phase shift on channel 1**

### 25.12.4 Multiple output signal combination

Each channel provides additionally the possibility to combine the channel inputs DTM_IN[x] and SHIFT[x] or OUT[x-1] (selected by I1SEL_[x]) by an AND or an XOR gate (selected by O1F_[x]).

It is recommended to use the combination of signals only if bit field RELBLK of register GTM0DTMiPSCTRL is 0. Otherwise, the signals TIM_CH_IN/DTM_AUX_IN may be disturbed by the blanking window counter.

Together with the inverter inside submodule DTM_PSU (selected by IN_POL), the inverter on each output of a channel (selected by POL0_[x]/POL1_[x]) and the possibility to change polarity of DTM_IN[x] inside connected TOM/ATOM channel, a (N)AND, (N)OR or X(N)OR combination of the signals is possible.

#### 25.12.4.1 Combination of input signal TIM_CH_IN/AUX_IN with TOM/ATOM signal

If the input selection I1SEL_[x] of a channel x is set to 0, the output selection O1SEL_[x] is set to 1 and SWAP_[x] is set to 0, depending on PSU_IN_SEL either TIM_CH_IN or DTM_AUX_IN can be combined with signal DTM_IN[x].

The function of combination on DTM output DTM[i]_OUT[x]_N (and also OUT[x]) is defined by O1F_[x] in the following way:

**Table 25.129 The function of combination on DTM output DTM[i]_OUT[x]_N**

| | O1F_x | POL1_x | IN_POL | (A)TOM output inverted |
|---|---|---|---|---|
| XOR | 01 | 0 | 0 | no |
| AND | 10 | 0 | 0 | no |
| XNOR | 01 | 1 | 0 | no |
| NAND | 10 | 1 | 0 | no |
| XNOR | 01 | 1 | 1 | yes |
| OR | 10 | 1 | 1 | yes |
| XOR | 01 | 0 | 1 | yes |
| NOR | 10 | 0 | 1 | yes |

**Note:** The inversion of the (A)TOM output can be reached by switching the **SL** bit (for TOM and ATOM SOMP/SOMC mode).

### 25.12.4.2 Combination of multiple TOM/ATOM output signals

If the input selection I1SEL_[x] of a channel x (with x = 1 to 3) is set to 1, the output selection O1SEL_[x] is set to 1 and SWAP_[x] is set to 0, the output of the preceding DTM channel OUT[x-1] can be combined with signal DTM_IN[x].

The function of combination on DTM output DTM[i]_OUT[x]_N (and also OUT[x]) is defined by O1F_[x] in the following way:

**Table 25.130 The function of combination on DTM output DTM[i]_OUT[x]_N**

| | O1F_x | POL1_x | POL1_x-1 | (A)TOM output inverted |
|---|---|---|---|---|
| XOR | 01 | 0 | 0 | no |
| AND | 10 | 0 | 0 | no |
| XNOR | 01 | 1 | 0 | no |
| NAND | 10 | 1 | 0 | no |
| XNOR | 01 | 1 | 1 | yes |
| OR | 10 | 1 | 1 | yes |
| XOR | 01 | 0 | 1 | yes |
| NOR | 10 | 0 | 1 | yes |

By setting I1SEL_[x] to 1 on all four channel, a combination of all four signals DTM_IN0 to DTM_IN3 can be achieved (combinatorial chain).

To allow also combination of signals generated for output DTM[i]_OUT[x], the outputs 0 and 1 can be swapped by setting bit SWAP_ [x] for channel x.

### 25.12.4.3 Pulse generation on edge

Another feature of the DTM is to generate on the second output DTM[i]_OUT[x]_N a pulse on every edge of corresponding input signal DTM[i]_IN[x].

This can be reached by configuring O1SEL_[x] to ‘1’, i.e. selecting signal edge_trigg_[x] as the output signal (O1F_[x] has to be '00'). The signal edge_trigg_[x] is depicted in **Figure 25.56, Wave signals for function of dead time generation**.

The pulse length can be adjusted individually for each edge type by the configuration value REL_RISE and REL_FALL of register DTM[i]_CH[x]_DV.

The parameter REL_RISE defines the pulse length in case of a rising edge on input DTM[i]_IN[x], the parameter REL_FALL define the pulse length in case of a falling edge on input DTM[i]_IN[x].

### 25.12.5 Synchronous update of channel control register 2

It is possible to use the shadow register GTM0DTMiCHCTRL2SR and a selected edge of one of the channel 0 to 3 to update the work register GTM0DTMiCHCTRL2.

The update mechanism and it's configuration is depicted in the following figure.

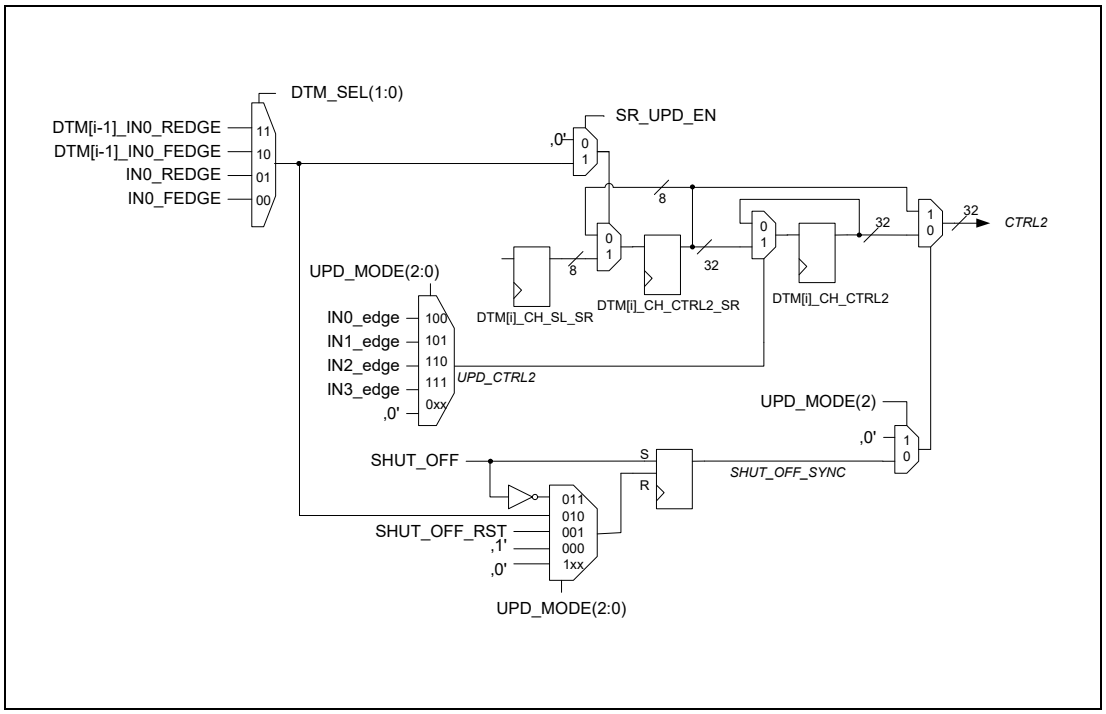



**Figure 25.60 Synchronous update mechanism of register GTM0DTMiCHCTRL2**

If enabled by the bit field UPD_MODE of register GTM0DTMiCTRL (i.e. UPD_MODE = 1xx), the register GTM0DTMiCHCTRL2SR serves as a shadow register of register GTM0DTMiCHCTRL2. The update is then triggered by an edge on one of the selected inputs DTM_IN0 to DTM_IN3.

The synchronous update allows the user to change output polarity, the selection of constant signal level, the constant signal level itself and the switch to/from feed through path on all four channels in parallel synchronized to one of the input edges on DTM_IN0 to DTM_IN3.

### 25.12.6 DTM output shut off

A fast shut off for the eight outputs of DTM instance i can be triggered by one of the two assigned inputs TIM[n]_CH_IN or DTM[i]_AUX_IN or the two inputs TIM[m]_CH_IN or DTM[i-1]_AUX_IN of the previous DTM instance i-1. (DTM_AUX_IN inputs are clamped to low (no function) for the P1x-C devices). The selection of the trigger signal source is done by the bits PSU_IN_SEL and DTM_SEL(1) (see **Figure 25.58** ). The selected trigger signal is named SHUT_OFF.

Enabling of the shut off feature is done by setting UPD_MODE(2:0) to one of the values “001”, “010” or “011”.

The shut off behavior of the DTM outputs is defined by the value of register GTM0DTMiCHCTRL2SR.

If the shut off feature is enabled by UPD_MODE, as long as the signal SHUT_OFF_SYNC is ‘0’, the register GTM0DTMiCHCTRL2 defines the output signal behavior.

If the signal SHUT_OFF_SYNC is ‘1’, the register GTM0DTMiCHCTRL2SR defines the output signal behavior.

The signal SHUT_OFF_SYNC is set to ‘1’ if signal SHUT_OFF switches to ‘1’ and is reset depending on value of UPD_MODE(2:0). The reset is higher prioritized than set

There are three different ways to reset the signal SHUT_OFF_SYNC to ‘0’:

- The CPU writes a ‘1’ to bit SHUT_OFF_RST of register DTM_CH_CTRL1
- Synchronous to an edge on DTM channel 0 input of this DTM instance i or on an edge on DTM channel 0 input of preceding DTM instance i-1.
- Asynchronous if signal SHUT_OFF switches back to '0'

**Figure 25.60** depicts the shut off feature and the different shut off release possibilities.

Note: The reset of SHUT_OFF_SYNC has higher priority then the set of this signal.

For the eight SL bits (SLx_y_SR) of the shadow register GTM0DTMiCHCTRL2SR exist a second shadow register DTM[i]_CH_SR.

If enabled by configuration bit SR_UPD_EN of register GTM0DTMiCTRL, the update of SL bits of register GTM0DTMiCHCTRL2SR can be triggered by one of the signals selected by bit field DTM_SEL of register GTM0DTMiCTRL. This trigger signal is either the rising or the falling edge detected on DTM[i]_IN0 of instance i or the rising or the falling edge on DTM[i-1]_IN0 of preceding instance i-1.

As depicted in **Figure 25.54** the DTM input signal TIM_CH_IN/DTM_AUX_IN can be forwarded to the succeeding instance. Thus, it can be used to trigger shut off in two consecutive DTM instances.

### 25.12.7 Configuration Register Overview

The following table gives an overview of the DTM configuration register.

**Table 25.131 Register list**

| Symbol | Register Name | Details in Section |
|---|---|---|
| GTM0DTMiCTRL | Global Configuration and Control Register | **25.12.8.1** |
| GTM0DTMiCHCTRL1 | Channel Control Register 1 | **25.12.8.2** |
| GTM0DTMiCHCTRL2 | Channel Control Register 2 | **25.12.8.3** |
| GTM0DTMiCHCTRL2SR | Channel Control Register 2 Shadow | **25.12.8.4** |
| GTM0DTMiPSCTRL | Phase Shift Unit Configuration and Control Register | **25.12.8.5** |
| GTM0DTMixDTV | Dead Time Reload Values; x = 0 to 3 | **25.12.8.6** |
| DTM[i]_CH_SR | Channel Shadow Register | **25.12.8.7** |

## 25.12.8 Configuration Register Description

### 25.12.8.1 GTM0DTMiCTRL (i = 24, 26, 28)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0DTM24CTRL: <GTM_base> + $13600_H$
GTM0DTM26CTRL: <GTM_base> + $13680_H$
GTM0DTM28CTRL: <GTM_base> + $13700_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SHUT_OFF_RST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SR_UPD_EN | — | UPD_MODE | | | DTM_SEL | | CLK_SEL | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.132 GTM0DTMiCTRL Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 16 | SHUT_OFF_RST | Shut off reset<br>Writing a '1' releases shut off (resets signal SHUT_OFF_SYNC if selected by UPD_MODE(2:0) = "001") |
| 15 to 9 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 8 | SR_UPD_EN | Shadow register update enable<br>0: No update of SLx_y_SR register bits in register GTM0DTMiCHCTRL2SR<br>1: Update of SLx_y_SR register bits in register GTM0DTMiCHCTRL2SR on trigger |
| 7 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 6 to 4 | UPD_MODE | Update mode<br>000: Asynchronous update – GTM0DTMiCHCTRL2SR not used for update of GTM0DTMiCHCTRL2<br>001: Shut off release by writing '1' to bit SHUT_OFF_RST of register GTM0DTMiCTRL<br>010: Shut off release by an edge on DTM[i]_IN0 or DTM[i-1]_IN0 (defined by bitfield DTM_SEL of register GTM0DTMiCTRL)<br>011: Shut off release by shut off signal SHUT_OFF (defined by bits PSU_IN_SEL and IN_POL of register GTM0DTMiPSCTRL and DTM_SEL(2) of register GTM0DTMiCTRL)<br>100: Signal IN0_edge used to trigger update of GTM0DTMiCHCTRL2 with content of GTM0DTMiCHCTRL2SR<br>101: Signal IN1_edge used to trigger update of GTM0DTMiCHCTRL2 with content of GTM0DTMiCHCTRL2SR<br>110: Signal IN2_edge used to trigger update of GTM0DTMiCHCTRL2 with content of GTM0DTMiCHCTRL2SR<br>111: Signal IN3_edge used to trigger update of GTM0DTMiCHCTRL2 with content of GTM0DTMiCHCTRL2SR |

**Table 25.132 GTM0DTMiCTRL Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3, 2 | DTM_SEL | Select DTM update and SHUT_OFF reset signal<br>00: Select falling edge on DTM[i] channel 0 input<br>01: Select rising edge on DTM[i] channel 0 input<br>10: Select falling edge on DTM[i-1] channel 0 input<br>11: Select rising edge on DTM[i-1] channel 0 input<br>0-: Shut off by signal TIM_CH_IN or DTM_AUX_IN<br>1-: Shut off by signal DTM[i-1]_PSU_IN |
| 1, 0 | CLK_SEL | Clock source select<br>00: SYS_CLK selected<br>01: CMU_CLK0 selected<br>10: CMU_CLK1 selected (if DTM is connected to an ATOM)/CMU_FXCLK0 selected (if DTM is connected to TOM)<br>11: CMU_CLK2 selected (if DTM is connected to an ATOM)/CMU_FXCLK1 selected (if DTM is connected to TOM) |

### 25.12.8.2 Register GTM0DTMiCHCTRL1 (i = 24, 26, 28)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0DTM24CHCTRL1: <GTM_base> + $13604_H$
GTM0DTM26CHCTRL1: <GTM_base> + $13684_H$
GTM0DTM28CHCTRL1: <GTM_base> + $13704_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | O1F_3 | | SWAP_3 | SH_EN_3 | I1SEL_3 | O1SEL_3 | — | XDT_EN_2_3 | O1F_2 | | SWAP_2 | SH_EN_2 | I1SEL_2 | O1SEL_2 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | O1F_1 | | SWAP_1 | SH_EN_1 | I1SEL_1 | O1SEL_1 | — | XDT_EN_0_1 | O1F_0 | | SWAP_0 | — | — | O1SEL_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R | R | R/W |

**Table 25.133 GTM0DTMiCHCTRL1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 29, 28 | O1F_3 | Output 1 function channel 3<br>00: Signal edge_trigg is selected<br>01: XOR of DTM[j]_IN3 and signal SHIFT3/OUT2<br>10: AND of DTM[j]_IN3 and signal SHIFT3/OUT2<br>11: DTM[j]_IN3_T selected |
| 27 | SWAP_3 | Swap outputs DTM[i]_CH[3]_OUT0 and DTM[i]_CH[3]_OUT1 (before final output register)<br>0: Outputs not swapped<br>1: Swap outputs DTM[i]_OUT3 and DTM[i]_OUT3_N |
| 26 | SH_EN_3 | Shift enable channel 3<br>0: DTM[i]_IN2 is not used; no input signal shift<br>1: Signal selected by I1SEL_3 triggers update of DTM[i]_IN3 with input of DTM[i]_IN2 -> input signal shift |
| 25 | I1SEL_3 | Input 1 select channel 3<br>0: Signal SHIFT2 selected<br>1: Signal OUT2 selected |
| 24 | O1SEL_3 | Output 1 select channel 3<br>0: Inverse dead time signal selected<br>1: Special function on output 1 selected (defined by O1F_3) |
| 23 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 22 | XDT_EN_2_3 | Cross dead time enable on channel 0 and 1<br>0: Cross dead time disabled on channel 2 and 3<br>1: Cross dead time enabled on channel 2 and 3 |
| 21, 20 | O1F_2 | Output 1 function channel 2<br>00: Signal edge_trigg is selected<br>01: XOR of DTM[j]_IN2 and signal SHIFT2/OUT1<br>10: AND of DTM[j]_IN2 and signal SHIFT2/OUT1<br>11: DTM[j]_IN2_T selected |
| 19 | SWAP_2 | Swap outputs DTM[i]_CH[2]_OUT0 and DTM[i]_CH[2]_OUT1 (before final output register)<br>0: Outputs not swapped<br>1: Swap outputs DTM[i]_OUT2 and DTM[i]_OUT2_N |

**Table 25.133 GTM0DTMiCHCTRL1 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 18 | SH_EN_2 | Shift enable channel 2<br>0: DTM[i]_IN1 is not used; no input signal shift<br>1: Signal selected by I1SEL_2 triggers update of DTM[i]_IN2 with input of DTM[i]_IN1 -> input signal shift |
| 17 | I1SEL_2 | Input 1 select channel 2<br>0: Signal SHIFT1 selected<br>1: Signal OUT1 selected |
| 16 | O1SEL_2 | Output 1 select channel 2<br>0: Inverse dead time signal selected<br>1: Special function on output 1 selected (defined by O1F_2) |
| 15, 14 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 13, 12 | O1F_1 | Output 1 function channel 1<br>00: Signal edge_trigg is selected<br>01: XOR of DTM[j]_IN1 and signal SHIFT1/OUT0<br>10: AND of DTM[j]_IN1 and signal SHIFT1/OUT0<br>11: DTM[j]_IN1_T selected |
| 11 | SWAP_1 | Swap outputs DTM[i]_CH[1]_OUT0 and DTM[i]_CH[1]_OUT1 (before final output register)<br>0: Outputs not swapped<br>1: Swap outputs DTM[i]_OUT1 and DTM[i]_OUT1_N |
| 10 | SH_EN_1 | Shift enable channel 1<br>0: DTM[i]_IN0 is not used; no input signal shift<br>1: Signal selected by I1SEL_1 triggers update of DTM[i]_IN1 with input of DTM[i]_IN0 -> input signal shift |
| 9 | I1SEL_1 | Input 1 select channel 1<br>0: Signal SHIFT1 selected<br>1: Signal OUT1 selected |
| 8 | O1SEL_1 | Output 1 select channel 1<br>0: Inverse dead time signal selected<br>1: Special function on output 1 selected (defined by O1F_1) |
| 7 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 6 | XDT_EN_0_1 | Cross dead time enable on channel 0 and 1<br>0: Cross dead time disabled on channel 0 and 1<br>1: Cross dead time enabled on channel 0 and 1 |
| 5, 4 | O1F_0 | Output 1 function channel 0<br>00: Signal edge_trigg is selected<br>01: XOR of DTM[j]_IN0 and signal SHIFT0<br>10: AND of DTM[j]_IN0 and signal SHIFT0<br>11: DTM[j]_IN1_T selected |
| 3 | SWAP_0 | Swap outputs DTM[i]_CH[0]_OUT0 and DTM[i]_CH[0]_OUT1 (before final output register)<br>0: Outputs not swapped<br>1: Swap outputs DTM[i]_OUT0 and DTM[i]_OUT0_N |
| 2, 1 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 0 | O1SEL_0 | Output 1 select channel 0<br>0: Inverse dead time signal selected<br>1: Special function on output 1 selected (defined by O1F_0) |

### 25.12.8.3 Register GTM0DTMiCHCTRL2 (i = 24, 26, 28)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0DTM24CHCTRL2: <GTM_base> + $13608_H$
GTM0DTM26CHCTRL2: <GTM_base> + $13688_H$
GTM0DTM28CHCTRL2: <GTM_base> + $13708_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DT1_3 | SL1_3 | OC1_3 | POL1_3 | DT0_3 | SL0_3 | OC0_3 | POL0_3 | DT1_2 | SL1_2 | OC1_2 | POL1_2 | DT0_2 | SL0_2 | OC0_2 | POL0_2 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DT1_1 | SL1_1 | OC1_1 | POL1_1 | DT0_1 | SL0_1 | OC0_1 | POL0_1 | DT1_0 | SL1_0 | OC1_0 | POL1_0 | DT0_0 | SL0_0 | OC0_0 | POL0_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.134 GTM0DTMiCHCTRL2 Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | DT1_3 | Dead time path enable on output 1 channel 3<br>0: Feed through from DTM_IN3 to DTM[i]_OUT3_N enabled<br>1: Dead time path enabled |
| 30 | SL1_3 | Signal level on output 1 channel 3<br>0: Signal Level is ‘0’ on output if OC1_3 = 1<br>1: Signal Level is ‘1’ on output if OC1_3 = 1 |
| 29 | OC1_3 | Output 1 control channel 3<br>0: Functional output<br>1: Constant output defined by SL1_3 |
| 28 | POL1_3 | Polarity on output 1 channel 3<br>0: Output signal not inverted<br>1: Output signal inverted |
| 27 | DT0_3 | Dead time path enable on output 0 channel 3<br>0: Feed through from DTM_IN3 to DTM[i]_OUT3 enabled<br>1: Dead time path enabled |
| 26 | SL0_3 | Signal level on output 0 channel 3<br>0: Signal Level is ‘0’ on output if OC0_3 = 1<br>1: Signal Level is ‘1’ on output if OC0_3 = 1 |
| 25 | OC0_3 | Output 0 control channel 3<br>0: Functional output<br>1: Constant output defined by SL0_3 |
| 24 | POL0_3 | Polarity on output 0 channel 3<br>0: Output signal not inverted<br>1: Output signal inverted |
| 23 | DT1_2 | Dead time path enable on output 1 channel 2<br>0: Feed through from DTM_IN2 to DTM[i]_OUT2_N enabled<br>1: Dead time path enabled |
| 22 | SL1_2 | Signal level on output 1 channel 2<br>0: Signal Level is '0' on output if OC1_2 = 1<br>1: Signal Level is '1' on output if OC1_2 = 1 |
| 21 | OC1_2 | Output 1 control channel 2<br>0: Functional output<br>1: Constant output defined by SL1_2 |

**Table 25.134 GTM0DTMiCHCTRL2 Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 20 | POL1_2 | Polarity on output 1 channel 2<br>0: Output signal not inverted<br>1: Output signal inverted |
| 19 | DT0_2 | Dead time path enable on output 0 channel 2<br>0: Feed through from DTM_IN2 to DTM[i]_OUT2 enabled<br>1: Dead time path enabled |
| 18 | SL0_2 | Signal level on output 0 channel 2<br>0: Signal Level is '0' on output if OC0_2 = 1<br>1: Signal Level is '1' on output if OC0_2 = 1 |
| 17 | OC0_2 | Output 0 control channel 2<br>0: Functional output<br>1: Constant output defined by SL0_2 |
| 16 | POL0_2 | Polarity on output 0 channel 2<br>0: Output signal not inverted<br>1: Output signal inverted |
| 15 | DT1_1 | Dead time path enable on output 1 channel 1<br>0: Feed through from DTM_IN1 to DTM[i]_OUT1_N enabled<br>1: Dead time path enabled |
| 14 | SL1_1 | Signal level on output 1 channel 1<br>0: Signal Level is '0' on output if OC1_1 = 1<br>1: Signal Level is '1' on output if OC1_1 = 1 |
| 13 | OC1_1 | Output 1 control channel 1<br>0: Functional output<br>1: Constant output defined by SL1_1 |
| 12 | POL1_1 | Polarity on output 1 channel 1<br>0: Output signal not inverted<br>1: Output signal inverted |
| 11 | DT0_1 | Dead time path enable on output 0 channel 1<br>0: Feed through from DTM_IN1 to DTM[i]_OUT1 enabled<br>1: Dead time path enabled |
| 10 | SL0_1 | Signal level on output 0 channel 1<br>0: Signal Level is '0' on output if OC0_1 = 1<br>1: Signal Level is '1' on output if OC0_1 = 1 |
| 9 | OC0_1 | Output 0 control channel 1<br>0: Functional output<br>1: Constant output defined by SL0_1 |
| 8 | POL0_1 | Polarity on output 0 channel 1<br>0: Output signal not inverted<br>1: Output signal inverted |
| 7 | DT1_0 | Dead time path enable on output 1 channel 0<br>0: Feed through from DTM_IN0 to DTM[i]_OUT0_N enabled<br>1: Dead time path enabled |
| 6 | SL1_0 | Signal level on output 1 channel 0<br>0: Signal Level is '0' on output if OC1_0 = 1<br>1: Signal Level is '1' on output if OC1_0 = 1 |
| 5 | OC1_0 | Output 1 control channel 0<br>0: Functional output<br>1: Constant output defined by SL1_0 |
| 4 | POL1_0 | Polarity on output 1 channel 0<br>0: Output signal not inverted<br>1: Output signal inverted |
| 3 | DT0_0 | Dead time path enable on output 0 channel 0<br>0: Feed through from DTM_IN0 to DTM[i]_OUT0 enabled<br>1: Dead time path enabled |
| 2 | SL0_0 | Signal level on output 0 channel 0<br>0: Signal Level is '0' on output if OC0_0 = 1<br>1: Signal Level is '1' on output if OC0_0 = 1 |

**Table 25.134 GTM0DTMiCHCTRL2 Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | OC0_0 | Output 0 control channel 0<br>0: Functional output<br>1: Constant output defined by SL0_0 |
| 0 | POL0_0 | Polarity on output 0 channel 0<br>0: Output signal not inverted<br>1: Output signal inverted |

### 25.12.8.4 GTM0DTMiCHCTRL2SR (i = 24, 26, 28)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0DTM24CHCTRL2SR: <GTM_base> + $1360C_H$
GTM0DTM26CHCTRL2SR: <GTM_base> + $1368C_H$
GTM0DTM28CHCTRL2SR: <GTM_base> + $1370C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DT1_3_SR | SL1_3_SR | OC1_3_SR | POL1_3_SR | DT0_3_SR | SL0_3_SR | OC0_3_SR | POL0_3_SR | DT1_2_SR | SL1_2_SR | OC1_2_SR | POL1_2_SR | DT0_2_SR | SL0_2_SR | OC0_2_SR | POL0_2_SR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DT1_1_SR | SL1_1_SR | OC1_1_SR | POL1_1_SR | DT0_1_SR | SL0_1_SR | OC0_1_SR | POL0_1_SR | DT1_0_SR | SL1_0_SR | OC1_0_SR | POL1_0_SR | DT0_0_SR | SL0_0_SR | OC0_0_SR | POL0_0_SR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.135 GTM0DTMiCHCTRL2SR Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | DT1_3_SR | Dead time path enable on output 1 channel 3 shadow register<br>0: Feed through from DTM_IN3 to DTM[i]_OUT3_N<br>1: Dead time path enabled |
| 30 | SL1_3_SR | Signal level on output 1 channel 3 shadow register<br>0: Signal Level is '0' on output if OC1_3 = 1<br>1: Signal Level is '1' on output if OC1_3 = 1 |
| 29 | OC1_3_SR | Output 1 control channel 3 shadow register<br>0: Functional output<br>1: Constant output defined by SL1_3 |
| 28 | POL1_3_SR | Polarity on output 1 channel 3 shadow register<br>0: Output signal not inverted<br>1: Output signal inverted |
| 27 | DT0_3_SR | Dead time path enable on output 0 channel 3 shadow register<br>0: Feed through from DTM_IN3 to DTM[i]_OUT3<br>1: Dead time path enabled |
| 26 | SL0_3_SR | Signal level on output 0 channel 3 shadow register<br>0: Signal Level is '0' on output if OC0_3 = 1<br>1: Signal Level is '1' on output if OC0_3 = 1 |
| 25 | OC0_3_SR | Output 0 control channel 3 shadow register<br>0: Functional output<br>1: Constant output defined by SL0_3 |
| 24 | POL0_3_SR | Polarity on output 0 channel 3 shadow register<br>0: Output signal not inverted<br>1: Output signal inverted |
| 23 | DT1_2_SR | Dead time path enable on output 1 channel 2 shadow register<br>0: Feed through from DTM_IN2 to DTM[i]_OUT2_N<br>1: Dead time path enabled |
| 22 | SL1_2_SR | Signal level on output 1 channel 2 shadow register<br>0: Signal Level is '0' on output if OC1_2 = 1<br>1: Signal Level is '1' on output if OC1_2 = 1 |
| 21 | OC1_2_SR | Output 1 control channel 2 shadow register<br>0: Functional output<br>1: Constant output defined by SL1_2 |

**Table 25.135 GTM0DTMiCHCTRL2SR Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 20 | POL1_2_SR | Polarity on output 1 channel 2 shadow register<br>0: Output signal not inverted<br>1: Output signal inverted |
| 19 | DT0_2_SR | Dead time path enable on output 0 channel 2 shadow register<br>0: Feed through from DTM_IN2 to DTM[i]_OUT2<br>1: Dead time path enabled |
| 18 | SL0_2_SR | Signal level on output 0 channel 2 shadow register<br>0: Signal Level is '0' on output if OC0_2 = 1<br>1: Signal Level is '1' on output if OC0_2 = 1 |
| 17 | OC0_2_SR | Output 0 control channel 2 shadow register<br>0: Functional output<br>1: Constant output defined by SL0_2 |
| 16 | POL0_2_SR | Polarity on output 0 channel 2 shadow register<br>0: Output signal not inverted<br>1: Output signal inverted |
| 15 | DT1_1_SR | Dead time path enable on output 1 channel 1 shadow register<br>0: Feed through from DTM_IN1 to DTM[i]_OUT1_N<br>1: Dead time path enabled |
| 14 | SL1_1_SR | Signal level on output 1 channel 1 shadow register<br>0: Signal Level is '0' on output if OC1_1 = 1<br>1: Signal Level is '1' on output if OC1_1 = 1 |
| 13 | OC1_1_SR | Output 1 control channel 1 shadow register<br>0: Functional output<br>1: Constant output defined by SL1_1 |
| 12 | POL1_1_SR | Polarity on output 1 channel 1 shadow register<br>0: Output signal not inverted<br>1: Output signal inverted |
| 11 | DT0_1_SR | Dead time path enable on output 0 channel 1 shadow register<br>0: Feed through from DTM_IN1 to DTM[i]_OUT1 enabled<br>1: Dead time path enabled |
| 10 | SL0_1_SR | Signal level on output 0 channel 1 shadow register<br>0: Signal Level is '0' on output if OC0_1 = 1<br>1: Signal Level is '1' on output if OC0_1 = 1 |
| 9 | OC0_1_SR | Output 0 control channel 1 shadow register<br>0: Functional output<br>1: Constant output defined by SL0_1 |
| 8 | POL0_1_SR | Polarity on output 0 channel 1 shadow register<br>0: Output signal not inverted<br>1: Output signal inverted |
| 7 | DT1_0_SR | Dead time path enable on output 1 channel 0 shadow register<br>0: Feed through from DTM_IN0 to DTM[i]_OUT0_N enabled<br>1: Dead time path enabled |
| 6 | SL1_0_SR | Signal level on output 1 channel 0 shadow register<br>0: Signal Level is '0' on output if OC1_0 = 1<br>1: Signal Level is '1' on output if OC1_0 = 1 |
| 5 | OC1_0_SR | Output 1 control channel 0 shadow register<br>0: Functional output<br>1: Constant output defined by SL1_0 |
| 4 | POL1_0_SR | Polarity on output 1 channel 0 shadow register<br>0: Output signal not inverted<br>1: Output signal inverted |
| 3 | DT0_0_SR | Dead time path enable on output 0 channel 0 shadow register<br>0: Feed through from DTM_IN0 to DTM[i]_OUT0 enabled<br>1: Dead time path enabled |
| 2 | SL0_0_SR | Signal level on output 0 channel 0 shadow register<br>0: Signal Level is '0' on output if OC0_0 = 1<br>1: Signal Level is '1' on output if OC0_0 = 1 |

**Table 25.135 GTM0DTMiCHCTRL2SR Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | OC0_0_SR | Output 0 control channel 0 shadow register<br>0: Functional output<br>1: Constant output defined by SL0_0 |
| 0 | POL0_0_SR | Polarity on output 0 channel 0 shadow register<br>0: Output signal not inverted<br>1: Output signal inverted |

### 25.12.8.5 Register GTM0DTMiPSCTRL (i = 24, 26, 28)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0DTM24PSCTRL: <GTM_base> + $13610_H$
GTM0DTM26PSCTRL: <GTM_base> + $13690_H$
GTM0DTM28PSCTRL: <GTM_base> + $13710_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | SHIFT_SEL | | — | — | IN_POL | PSU_IN_SEL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RELBLK | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.136 GTM0DTMiPSCTRL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 21, 20 | SHIFT_SEL | Shift select<br>00: DTM channel 1 is connected via signal SHIFT1 with TIM_CH_IN/ DTM_AUX_IN<br>01: DTM channel 2 is connected via signal SHIFT2 with TIM_CH_IN/ DTM_AUX_IN<br>10: DTM channel 3 is connected via signal SHIFT3 with TIM_CH_IN/ DTM_AUX_IN<br>11: DTM channel 0 is connected via signal SHIFT0 with TIM_CH_IN/ DTM_AUX_IN |
| 19, 18 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 17 | IN_POL | Input polarity<br>0: Input signal is not inverted<br>1: Input signal is inverted |
| 16 | PSU_IN_SEL | PSU input select<br>0: TIM_CH_IN selected<br>1: DTM_AUX_IN selected |
| 15 to 10 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 9 to 0 | RELBLK | Reload value blanking window<br>**NOTE**<br>A value of $000_H$ resets counter BLK_DOWN_CNT |

### 25.12.8.6 Register GTM0DTMixDTV (i = 24, 26, 28, x = 0 to 3)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0DTM240DTV: <GTM_base> + $13614_H$, GTM0DTM241DTV: <GTM_base> + $13618_H$
GTM0DTM242DTV: <GTM_base> + $1361C_H$, GTM0DTM243DTV: <GTM_base> + $13620_H$
GTM0DTM260DTV: <GTM_base> + $13694_H$, GTM0DTM261DTV: <GTM_base> + $13698_H$
GTM0DTM262DTV: <GTM_base> + $1369C_H$, GTM0DTM263DTV: <GTM_base> + $136A0_H$
GTM0DTM280DTV: <GTM_base> + $13714_H$, GTM0DTM281DTV: <GTM_base> + $13718_H$
GTM0DTM282DTV: <GTM_base> + $1371C_H$, GTM0DTM283DTV: <GTM_base> + $13720_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RELFALL | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RELRISE | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.137 GTM0DTMixDTV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 25 to 16 | RELFALL | Reload value for falling edge dead time |
| 15 to 10 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 9 to 0 | RELRISE | Reload value for rising edge dead time |

### 25.12.8.7 Register DTM[i]_CH_SR (i = 24, 26, 28)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0DTM24CHSR: <GTM_base> + $13624_H$
GTM0DTM26CHSR: <GTM_base> + $136A4_H$
GTM0DTM28CHSR: <GTM_base> + $13724_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | SL1_3_SR_SR | SL0_3_SR_SR | SL1_2_SR_SR | SL0_2_SR_SR | SL1_1_SR_SR | SL0_1_SR_SR | SL1_0_SR_SR | SL0_0_SR_SR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.138 DTM[i]_CH_SR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 7 | SL1_3_SR_SR | Shadow register for bit SL1_3_SR of register GTM0DTMiCHCTRL2SR |
| 6 | SL0_3_SR_SR | Shadow register for bit SL0_3_SR of register GTM0DTMiCHCTRL2SR |
| 5 | SL1_2_SR_SR | Shadow register for bit SL1_2_SR of register GTM0DTMiCHCTRL2SR |
| 4 | SL0_2_SR_SR | Shadow register for bit SL0_2_SR of register GTM0DTMiCHCTRL2SR |
| 3 | SL1_1_SR_SR | Shadow register for bit SL1_1_SR of register GTM0DTMiCHCTRL2SR |
| 2 | SL0_1_SR_SR | Shadow register for bit SL0_1_SR of register GTM0DTMiCHCTRL2SR |
| 1 | SL1_0_SR_SR | Shadow register for bit SL1_0_SR of register GTM0DTMiCHCTRL2SR |
| 0 | SL0_0_SR_SR | Shadow register for bit SL0_0_SR of register GTM0DTMiCHCTRL2SR |

# 25.13 Multi Channel Sequencer (MCS)

## 25.13.1 Overview

The Multi Channel Sequencer (MCS) sub module is a generic data processing module that is connected to the ARU. One of its major applications is to calculate complex output sequences that may depend on the time base values of the TBU and are processed in combination with the ATOM sub module. Other applications can use the MCS sub module to perform extended data processing of input data resulting from the TIM sub module. Moreover, some applications may process data provided by the CPU within the MCS sub module, and the calculated results are sent to the outputs using the ATOM sub modules.

The following parameters are design variables for the MCS hardware structure that can vary in its range for different module instances:

W - Word width of the data path

T - Number of available MCS channels

RDW - RAM data width of connected RAM

RAW - RAM address width used by the MCS for addressing memory

NPS - Number of pipeline stages

USR - Use second RAM (0 - one RAM available, 1 – two RAMs available)

All MCS instances in the GTM use the values W = 24, RDW = 32, and NPS = 5. The values for T and RAW are device specific and vary between the following values:

T = 3, 6, or 9

RAW = 9

USR = 0 or 1.

## 25.13.2 Architecture

### 25.13.2.1 MCS Architecture

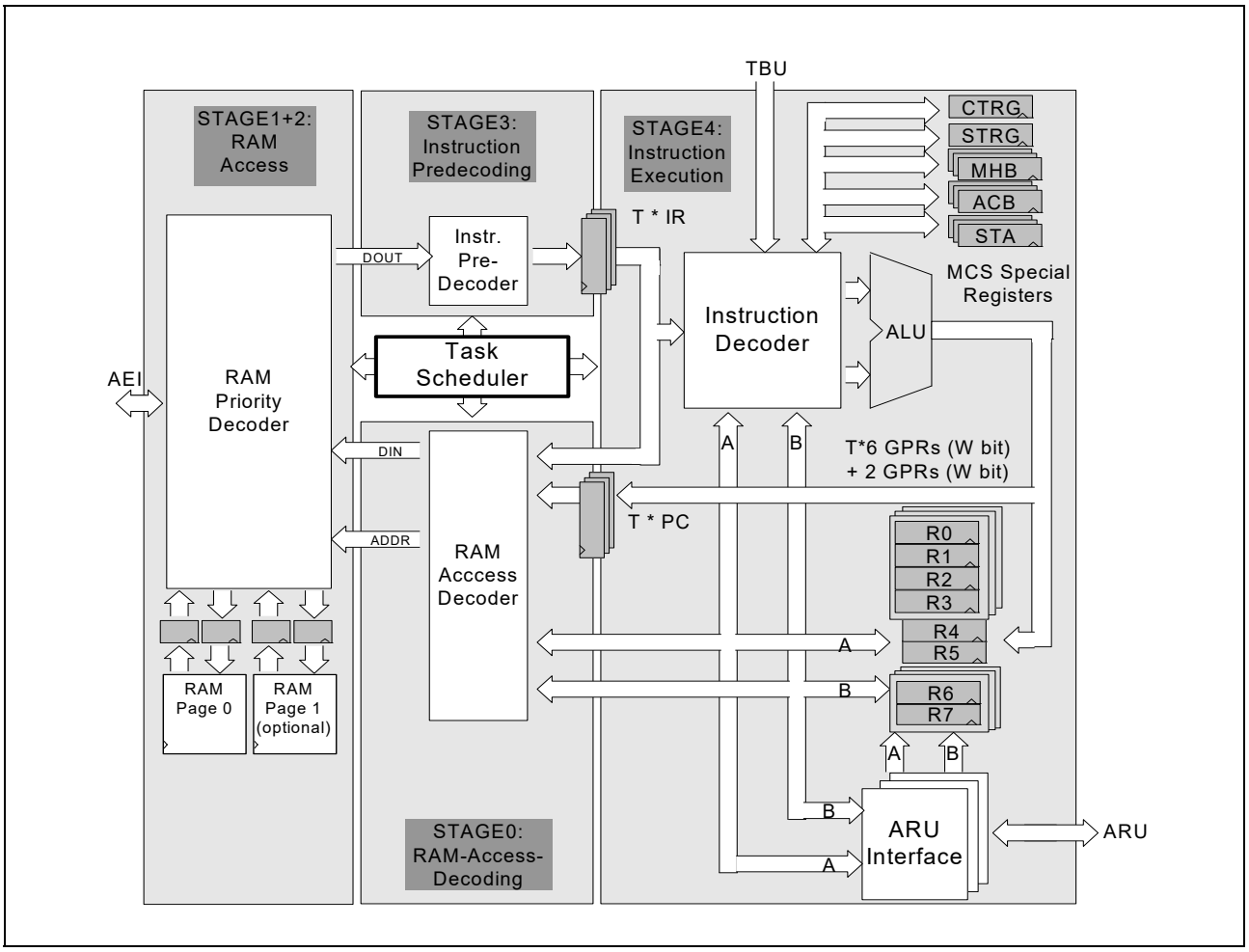



Figure 25.61 MCS Architecture block diagram

**Figure 25.61** gives an overview of the MCS architecture.

The MCS module mainly embeds a single data path with five pipeline stages, consisting of a W-bit wide Arithmetic Logic Unit (ALU), several decoders, and a connection to one or two RAM modules (depending on parameter USR) located outside of the MCS sub module.

The data path of the MCS is shared by T so called MCS-channels, whereas each MCS-channel executes a dedicated micro-program that is stored inside the RAM connected to the MCS module.

The connected RAM may contain arbitrary sized code and data sections that are accessible by all MCS-channels and the CPU via AEI bus interface. More details about the RAM can be found in **Section 25.13.4, Memory Organization**

An MCS-channel can also be considered as an individual task of a processor that is scheduled to the commonly used data path at a specific point in time. The execution of the different MCS-channels on the different pipeline stages is controlled by a central hardware related task scheduler, which enables fast task switches within a single clock cycle. Details about the task scheduler and the available scheduling algorithms can be found in **Section 25.13.3, Scheduling**.

Typically, if data has to be exchanged between different MCS-channels and/or the CPU, the connected RAM, which is accessible by all MCS-channels and the CPU, can be used.

Besides the commonly used data path, each MCS-channel has a dedicated ARU interface for communication with other ARU connected modules, an Instruction Register (IR), a Program Counter Register (PC), a Status Register (STA), an ARU Control Bit Register (ACB), a Memory High Byte Register (MHB) and a Register Bank with eight W bit general purpose registers (R0, R1, ....R7). With exception to the registers R4 and R5, all of these registers are only visible within its dedicated MCS-channel and thus the MCS-channels cannot exchange data using registers. The registers R4 and R5 are common to all MCS-channels and thus they can also be used for sharing data between MCS-channels. Moreover, the general purpose registers (R0, R1, ... R7) are also writable by the CPU, an thus an MCS channel may also consume data from the CPU using its general purpose registers.

The MCS also provides a common 16 bit wide trigger register that can be accessed by all MCS channels in order to trigger other MCS-channels located in the same module. Writing to STRG sets bits and writing to CTRG clears bits in the common trigger register. To enable triggering of MCS-channels by CPU, the CPU can set bits in the common trigger register by writing to GTM0MCSiSTRG and clear bits by writing to GTM0MCSiCTRG.

Considering the architecture in the figure above, the main actions of the different pipeline stages are as follows:

Pipeline stage 0 performs a setup of address, input data, and control signals for the next RAM access of a specific MCS-channel.

The actual RAM access of a specific MCS-channel is executed in pipeline stage 1 and 2, assuming an external connection of a synchronous RAM with a latency of one clock cycle.

Pipeline stage 3 performs pre-decoding and dispatching of instructions and data resulting from the RAM.

Finally, in pipeline stage 4 the current instruction is executed.

Since the internal registers of the MCS can be updated by different sources (MCS write access, CPU write access, ARU read access) a write conflict occurs if more than one source wants to write to the same register. In this case the result of the register is unpredictable. However, the software should setup its application in a way that such conflicts do not occur.

One exception is the common trigger register, which may be written by multiple sources (different MCS channels and CPU) in order to enable triggering of different MCS channels. Typically, the software should setup its application in a manner that different sources should not write the same bits in the trigger register.

### 25.13.3 Scheduling

The MCS provides a hardware related task scheduler, which globally controls the execution of the tasks in the different pipeline stages. The task scheduler implements four different scheduling modes, that can be selected by the SCD_MODE bit field in the GTM0MCSiCTRLSTAT register. Depending on the selected scheduling mode, the task scheduler is selecting a dedicated MCS channel that will be executed in pipeline stage 0 in the next clock cycle. Additionally, MCS channels that are already present in the pipeline are shifted to its successor pipeline stage, with each clock cycle. This means, that the execution time of an MCS-channel in a specific pipeline stage is always one clock cycle.

The MCS task scheduler may also schedule an empty cycle to pipeline stage 0, in order to grant a time slice to the CPU for accessing the connected RAM.

It should be noted, if the task scheduler assigns an MCS-task to pipeline stage 0, but this task does not access the RAM, the CPU can access the corresponding RAM, even if the scheduler did not reserve an empty clock cycle.

In the following, the available scheduling modes are described.

#### 25.13.3.1 Round Robin Scheduling

The Round Robin Scheduling Mode implements the simplest scheduling algorithm. This algorithm schedules a predefined set of MCS channels in the range [0; SCD_CH] in ascending order. After the last channel SCD_CH has been assigned to the pipeline, an empty cycle is scheduled in order to enable RAM access for the CPU. The parameter SCD_CH can be controlled by the register GTM0MCSiCTRLSTAT. If the value of SCD_CH is greater than T-1, the scheduler assumes a value of T-1 for bit field SCD_CH.

**Figure 25.62** shows a timing example of the Round Robin Scheduling with T = 8 MCS-channels (marked as $C_0$ to $C_7$) that are scheduled together with a CPU access to a pipeline with and NPS = 5 stages. It is assumed that bit field SCD_CH is set to 8.

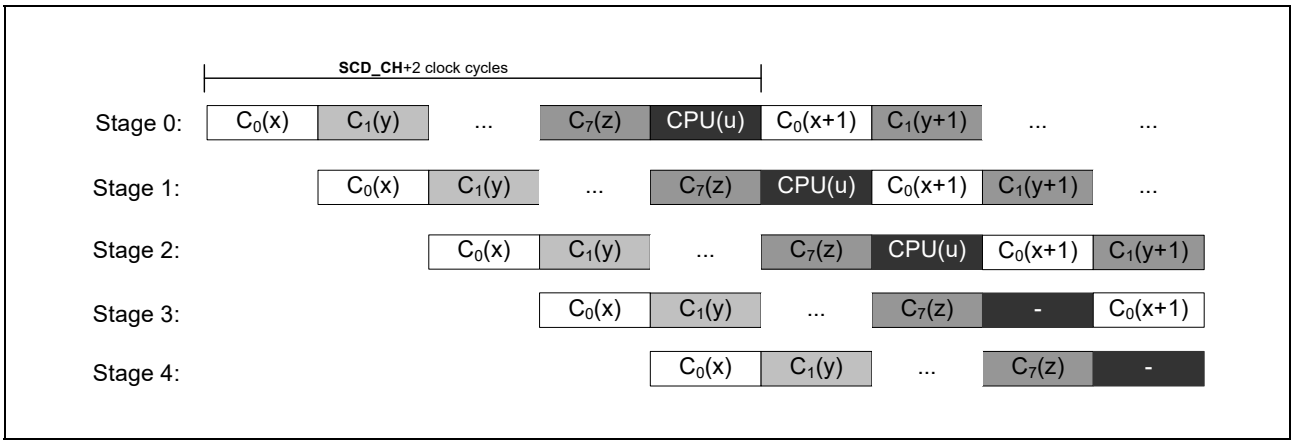



Figure 25.62 Timing of Round Robin Scheduling

The identifier Ci(x) denotes that MCS-channel i is currently executing the instruction or data located in the memory at position x in the corresponding pipeline stage. The figure shows, which MCS-channel is activated in specific pipeline stage at a specific point in time.

Moreover, the figure shows, that the Round Robin scheduling is always repeated after SCD_CH+2 clock cycles, which means that the time duration of an instruction cycle is SCD_CH+2 clock cycles. However, if the value SCD_CH + 2 is less than NPS, the duration of an instruction cycle is limited by

the depth of the pipeline to NPS clock cycles. Thus the effective execution time of a single cycle instruction is always MIN(SCD_CH+2, NPS) clock cycles, ignoring the latency of the pipeline.

The Round Robin scheduling algorithm has the characteristic that it fairly distributes all time slices to all MCS-channels and the CPU. This means, that the program execution time of a specific task is independent from the activity of any neighboring task or the CPU RAM access, and thus a correct estimation of the actual program execution time is very easy. However, the round-robin scheduling may waste clock cycles by scheduling MCS-channels that are not ready to execute an instruction (e.g. MCS-channel is disabled by CPU). The following scheduling modes overcome this issue.

#### 25.13.3.2 Accelerated Scheduling

In order to improve the computational performance, the accelerated scheduling mode provides two key features. Firstly, the scheduler only selects MCS-channels that are not suspended and thus can actually execute an instruction. Secondly, the scheduler applies instruction prefetching to minimize empty cycles in the pipeline. An MCS-channel is marked as suspended due to one of the reasons:

- An MCS-channel is executing a read or write request to an ARU connected sub module (instruction ARD, AWR, ARDI, AWRI, NARD, NARDI).
- An MCS-channel waits on a register match event (e.g. instruction WURM), in order to wait on a desired register value (e.g. trigger event from another MCS channel).
- An MCS-channel is disabled.

In the case of instruction prefetching, the scheduler will assign an MCS-channel $C_p$ to pipeline stage 0, which is already present in another pipeline stage. This means, that the execution of the last instruction of $C_p$ located in the memory MEM(PC/4) is not yet finished completely, whereas PC is the current value of the program counter of MCS-channel $C_p$. Thus, the newly scheduled MCS-channel $C_p$ will prefetch a successor instruction MEM(PC/4+PFO) under the assumption that there will be no branch and no memory access in the program between the instructions MEM(PC/4) and MEM(PC/4+PFO). The prefetch offset value PFO is determined by counting the number of already scheduled MCS channels $C_p$ in the pipeline. However, if the assumption fails, the pipeline will be flushed by replacing all MCS-channel $C_p$ of the pipeline with an empty cycle, as soon as the instruction decoder detects a branch or a memory access. In general, each MCS-channel can accept instruction prefetching. However, there are some cases in which an upcoming flushing of the pipeline can be easily detected by the MCS hardware due to evaluation of internal states. Therefore, it is defined that an MCS-channel accepts instruction prefetching only under the following conditions:

- An MCS-channel is currently not in the second cycle of a two-cycle control flow instruction (instruction CALL, RET).
- An MCS-channel is currently not in the second cycle of a three-cycle memory access instruction (instruction MWRL, MWRIL).

The accelerated scheduling mode guarantees, that the time duration of an instruction cycle varies between 1 and T+1 cycles. Hence, a single cycle instructions has an effective execution time between 1 to T+1 clock cycles, depending on the number of suspended MCS-channels and the actual instruction sequence. The worst case execution time occurs if all channels are active and the CPU also accesses the RAM. The best case occurs e.g. if only one MCS-channel is enabled and the executed program sequence has only linear code without branches and memory access.

The algorithm of the accelerated scheduling mode first, evaluates the state of all available MCS-channels as well as a CPU request to the RAMs and then it decides if a specific MCS-channel or an empty cycle is assigned to pipeline stage 0 in the next clock cycle. It should be noted that the accelerated scheduling mode treats RAM access requests from the CPU in a similar manner as MCS-

channels, which means that empty cycles for RAM requests are only inserted into the pipeline if there is an active RAM request from the CPU or no other task can be scheduled.

In order to fairly trade all available MCS-channels as well as CPU RAM requests and to guarantee a worst case execution time of T+1 clock cycles, an additional task prioritization scheme is applied used that dynamically prioritizes all MCS-channels and a CPU memory access depending on the history of the scheduler's decisions. The algorithm of the accelerated scheduler mode is executed every clock cycle and it works in the following manner:

1. Try to find an MCS-channel $C_r$ with highest priority that is not suspended and not already scheduled to the pipeline stages 0 to NPS-2. If $C_r$ is found assign $C_r$ to pipeline stage 0 and finish scheduling for current clock cycle.
2. Otherwise, try to find an MCS-channel $C_p$ with highest priority that is not suspended and accepts instruction prefetching. If $C_p$ is found assign $C_p$ to pipeline stage 0 and finish scheduling for current clock cycle.
3. Otherwise, try to find an MCS-channel $C_s$ with highest priority that is suspended and accepts instruction prefetching. If $C_s$ is found assign $C_s$ to pipeline stage 0 and finish scheduling for current clock cycle.
4. Otherwise, assign an empty cycle to pipeline stage 0 and finish scheduling for current clock cycle.

The underlying task prioritization scheme tracks the history of the scheduled MCS-channels in a list consisting of T+1 items. The list is initialized with all MCS-channels followed by a reserved time slot for the CPU RAM access. The position of an MCS-channel within this list implicitly defines the priority, while the back of this list holds the MCS-channel with highest priority. Whenever the scheduling algorithm described above has found an MCS-channel $C_r$ or $C_p$ it to be scheduled in the next clock cycle, it removes this item from the list and put it to the front of the list. In order to fairly prioritize all MCS-channels, the algorithm also removes the item at the back of the list to the second position in the list, after the inserted scheduled front item. Since the list always contains all possible MCS-channels and with each clock cycles each non-scheduled item is moved at least one position towards the end of list, it is obvious that each MCS-channel will have the highest priority not later than T+1 clock cycles.

**Figure 25.63** shows a timing example of the accelerated scheduling with NPS=5 pipeline stages.

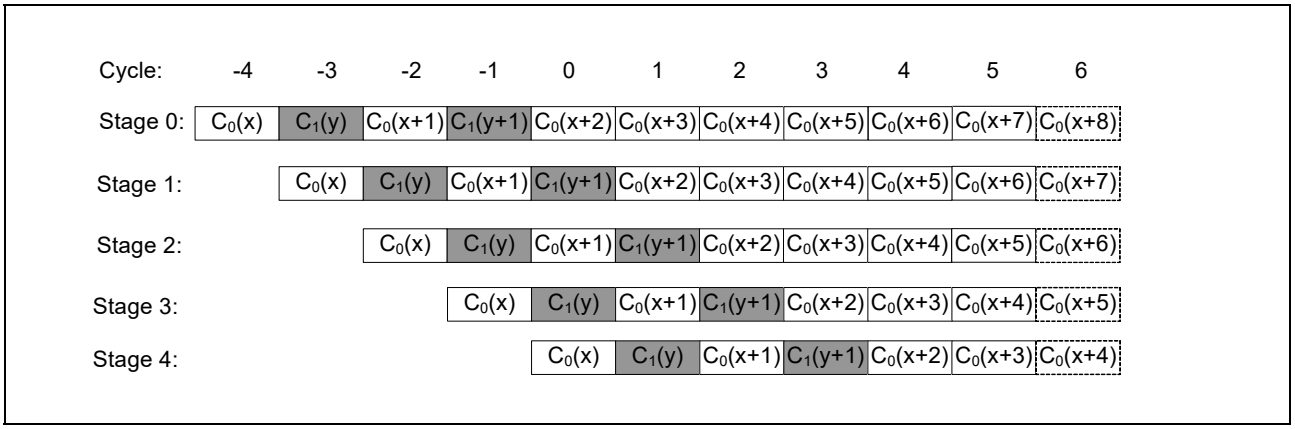



**Figure 25.63 Timing of Accelerated Scheduling**

The example assumes that initially MCS-channels 0 and 1 are enabled and the program for each MCS-channel is located in the RAM as follows:

**MCS-Channel 0**

| Memory Location | Instruction |
|---|---|
| x+0 | ADDL R0, 7 |
| x+1 | XOR R0, R1 |
| x+2 | SHR R0, 7 |
| x+3 | JBC STA, Z, 4*(x+9) |
| x+4 | MOVL R2, 5 |

**MCS-Channel 1**

| Memory Location | Instruction |
|---|---|
| y+0 | MOVL STA, 0 |

**Figure 25.64 The example assumes that initially MCS-channels 0 and 1**

Since both channels are ready to run, the scheduler fairly selects the channels in an alternating order, as it can be obtained in stage 0 at the clock cycles before cycle 1. Since MCS-channel 1 is disabling itself at cycle 1 with the instruction of memory location y, the scheduler will only select MCS-channel 0 in the following by applying instruction prefetching. But it should be noted, the scheduler applies instruction prefetching during the whole sequence, due to the fact that the number of enabled channels is always less than the available number of pipeline stages NPS. The actual state of pipeline in cycle 6 depends on conditional branch instruction of memory location x + 3. If the branch is not taken, the linear code execution of MCS-channel 0 is continued as shown in the Figure. However, if the branch to memory location x+9 is taken, in cycle 6 the scheduler will fetch the instruction $C_0$(x+9) in stage 0 and flush the stages 1 to NPS-1. Note, the flushing of the pipeline only concerns the prefetched instructions of the MCS-channel that is currently executed in the last stage. If pipeline stage 1 of cycle 5 would belong to another channel than 0, only the stages greater than 2 would be flushed.

#### 25.13.3.3 Single Prioritization Scheduling

The Single Prioritization Scheduling mode is an extended variant of the Accelerated Scheduling mode, which additionally applies a task prioritization of a single MCS-channel. In this mode, the bit field SCD_CH of register GTM0MCSiCTRLSTAT is used to identify a dedicated MCS-channel that is always preferred during scheduling. This means, that the scheduler will assign preferred MCS-channel SCD_CH to pipeline stage 0, as long as this channel is not suspended. If the preferred MCS-channel is entering its suspended state, the scheduling algorithm switches to the accelerated scheduling as previously described in **Section 25.13.3.2, Accelerated Scheduling**. Whenever the MCS-channel SCD_CH is resuming from its suspended state, the scheduler switches back and assign the channel SCD_CH to pipeline stage 0 until the next suspension event occurs. If the bifield SCD_CH contains the value T or higher, the task scheduler will always prioritize CPU access to the RAM. This means, whenever the task scheduler detects that the CPU wants to access an MCS-RAM, the scheduler will assign an empty cycle into pipeline stage 0. If the CPU does not access the RAM any more, it switches back to the accelerated mode, as described previously in **Section 25.13.3.2, Accelerated Scheduling**.

In consequence, the Single Prioritization Scheduling mode cannot guarantee a maximum time duration of an instruction cycle for the overall execution of all MCS-channels, since it strongly depends on the activity of the prioritized MCS-channel SCD_CH. However, the Single Prioritization Scheduling mode provides the fastest possible execution for MCS-channel SCD_CH. Moreover, during the time spawn, in which the prioritized MCS-channel SCD_CH is suspended, this mode guarantees a duration of 1 to T+1 clock cycles of an instruction cycle for all non-prioritized channels.

#### 25.13.3.4 Multiple Prioritization Scheduling

The Multiple Prioritization Scheduling mode is an extended variant of the Accelerated Scheduling mode, which additionally applies a task prioritization for multiple MCS-channels. In this mode, the bit field SCD_CH of register GTM0MCSiCTRLSTAT is used to identify a set of dedicated MCS-channels, which are always preferred during scheduling. The identifiers of the prioritized MCS-channels are in the range [0; SCD_CH] and the non-prioritized channels are in the range [SCD_CH+1; T-1]. The individual priority for the set of prioritized MCS-channels is applied in descending order, which means that MCS-channel 0 has the highest priority, followed MCS-channel 1, which has the second highest priority, and so on. The non-prioritized MCS-channels do not have any priority. A value of T-1 or higher for the bifield SCD_CH means that all T MCS-channels are prioritized MCS-channels.

With each clock cycle, the Multiple Prioritization Scheduling mode will assign the non-suspended MCS-channel with the highest priority from the set of prioritized MCS-channels to pipeline stage 0, as long as there are non-suspended prioritized MCS-channels available. If all prioritized MCS-channels are suspended, the scheduling algorithm switches to the accelerated scheduling as previously described in **Section 25.13.3.2, Accelerated Scheduling** and it schedules the non-prioritized channels. Whenever a prioritized MCS-channel is resuming from its suspended state, the scheduler switches back and applies the described prioritization scheme until the next suspension event of occurs.

In consequence, the Multiple Prioritization Scheduling mode cannot guarantee a maximum time duration of an instruction cycle for the overall execution of all MCS-channels, since it strongly depends on the activity of the prioritized MCS-channels. However, the Multiple Prioritization Scheduling mode provides the fastest possible execution for prioritized MCS-channels. Moreover, during the time spawn, in which all prioritized MCS-channels are suspended, this mode guarantees a duration of 1 to T+1 clock cycles of an instruction cycle for all non-prioritized channels.

### 25.13.4 Memory Organization

The MCS module supports a memory layout of up to $2^{RAW+USR}$ memory locations each RDW bit wide leading to a maximum byte wise address range from 0 to $2^{RAW+USR+2}-1$.

If two RAM modules are used (USR = 1) the entire address space of the MCS is divided into two seamless memory pages. Further, if the GTM provides a memory configuration sub module (MCFG), memory page 0 begins from (byte wise) address 0 and ranges to address MP0-4 and memory page 1 ranges from MP0 to MP1-4, while MP0 and MP1 are configuration parameters provided by MCFG. If USR is 1 but there is no MCFG module available, the actual parameters MP0 and MP1 can be found in [1]. The actual for addresses for accessing the memories via AEI can also be found in [1].

The RAM priority decoder of the MCS will always handle a RAM access from an MCS channel with a higher priority compared to a RAM access from AEI.

However, if a set of active MCS channels are only accessing one common RAM page, the MCS will grant any AEI accesses to the other RAM page in parallel to the MCS related RAM accesses, which means that AEI may get the full bandwidth to a dedicated RAM.

Basically, the actual access time to the RAMs via AEI depends on the actual scheduling mode and the activity of tasks. In the modes Round Robin Scheduling and Accelerated Scheduling the scheduler guarantees a maximum write access time of T + 3 clock cycles and a maximum read access time of T+ 6 clock cycles (assuming that the AEI bridge is configured with standard protocol in synchronous mode). In the scheduling modes Single Prioritization Scheduling and Multiple Prioritization Scheduling, the scheduler cannot guarantee a maximum access time for AEI RAM access.

Depending on the silicon vendor configuration, the connected RAM pages are initialized with zeros in the case of an MCS module reset.

If an ECC Error occurs while an MCS-channel reads data from a memory module, the corresponding MCS-channel is disabled and the ERR bit in register STA is raised.

### 25.13.5 Instruction Set

This section describes the entire instruction set of the MCS sub module. First, a brief overview over all available instructions is given and a detailed description of each instruction can be found in **Section 25.13.5.2, Data Transfer Instructions** to **Section 25.13.5.7, Other Instructions**.

In general, each instruction is RDW bit wide but the duration of each instruction varies between several instruction cycles. As already described in **Section 25.13.3, Scheduling** , the number of required clock cycles for an instruction cycle can be fixed or variable, depending on the selected scheduling mode. In the case of the Round Robin Scheduling, the duration is fixed with T+1 clock cycles, in the case of the Accelerated Scheduling the duration is variable in the range between 1 and T+1 clock cycles, and in all other Scheduling modes the duration is also variable and may even be more than T+1 clock cycles, depending on the application.

Before the available instructions are described, some commonly used terms, abbreviations and expressions are introduced:

OREG: The operation register set OREG = {R0, R1, R2, ..., R7, STA, ACB, CTRG, STRG, TBU_TS0, TBU_TS1, MHB} includes all MCS accessible internal registers, as well as the global time bases TBU_TS0, and TBU_TS1 that are provided by the sub module TBU.

AREG: The ARU register set AREG = {R0, R1, R2, ..., R7, ZERO} includes the all registers that can be written by incoming ARU transfers (ARD, ARDI, NARD, and NARDI instructions). These registers include all eight general purpose registers. The dummy register ZERO may be used to discard an incoming 24 bit ARU word.

**NOTE**

In the following, the register sets OREG and AREG are referred by the instructions. Typically, an operation announces W data bits. Whenever, a register of OREG implements less than W bits, it is assumed that these register bits only define the lower significant bits of an operation. The missing most significant bits are always read and written as zeros.

WLIT: The set WLIT = {0,1, ..., $2^{W}$-1} is a W bit wide literal value used for encoding immediate operands.

ALIT: The set ALIT = {0,1, ..., $2^{RAW+USR}$-1} is a RAW + USR bit wide literal value used for encoding memory addresses.

AOLIT: The set AOLIT = {-$2^{RAW+USR-1}$, ..., -1,0,1, ..., $2^{RAW+USR-1}$-1} is a RAW + USR bit wide literal value used for encoding relative memory address offsets.

ARDLIT: The set ARDLIT = {0,1, ..., $2^{9}$-1} is a 9 bit literal used for ARU read addresses.

AWRLIT: The set AWRLIT = {0,1, ..., 23} is used as ARU write indexes, selecting one of the 24 ARU write address.

SFTLIT: The set SFTLIT = {0,1, ..., W} is used as literal value for shift instructions.

BITLIT: The set BITLIT = {0,1, ..., 15} is a 4 bit literal used for bit indexing.

MSKLIT: The set MSKLIT = {0,1, ..., $2^{15}$-1} is a 16 bit literal used for bit-masking.

BIT SELECTION: The expression VAR[i] represents the i-th bit of a variable VAR.

BIT RANGE SELECTION: The expression VAR[m:n] represents the bit slice of variable VAR that is ranging from bit n to bit m.

MEMORY ADRESSING: The expression MEM(X) represents the RDW bit wide value at location x (x ∈ ALIT) of the memory. The expression MEM(x)[m:n] represents the bit slice ranging from bit n to m of the RDW bit wide word at memory location x.

ARU ADRESSING: In the case of ARU reading, the expression ARU(x) represents the 2*W+5 bit wide ARU word of ARU channel at read address x (x ∈ ARDLIT). In the case of ARU writing, the expression ARU(x) represents a 2*W+5 bit wide ARU word that is written to an ARU channel indexed by the index x (x ∈ AWRLIT). The index x selects a single ARU write channel from the pool of the MCS sub module's allocated ARU write channels. An MCS sub module has 24 dedicated ARU write channels, indexed by values 0 to 23. The expression ARU(x)[m:n] represents the bit slice ranging from bit n to m of the 2*W+5 bit wide ARU word.

**Table 25.139** summarizes the entire instruction set of the MCS and **Table 25.140** shows the encoding of the individual instructions.

Table 25.139 Instruction Set Summary (1/3)

| Class | Mnemonic | Operation | Instruction cycles | Synopsis |
|---|---|---|---|---|
| Data transfer | MOVL A, C | A ← C | 1 | Move Literal, A in OREG, C in WLIT |
| | MOV A, B | A ← B | 1 | Move, A in OREG, B in OREG |
| | MRD A, C | A ← MEM(C)[W-1:0];<br>MHB ← MEM(C)[RDW-1:W] | 2*1 | Memory Read, A in OREG, C in ALIT |
| | MWR A, C | MEM(C)[W-1:0] ← A;<br>MEM(C)[RDW-1:W] ← MHB | 2*1 | Memory Write, A in OREG, C in ALIT |
| | MRDI A, B [, C] | A ← MEM(B[RAW+USR+1:2]+C)[W-1:0];<br>MHB ← MEM(B[RAW+USR+1:2]+C)[RDW-1:W] | 2*1 | Memory Read Indirect, A in OREG, B in OREG, C in AOLIT |
| | MWRI A, B [, C] | MEM(B[RAW+USR+1:2]+C)[W-1:0] ← A;<br>MEM(B[RAW+USR+1:2]+C)[RDW-1:W] ← MHB | 2*1 | Memory Write Indirect, A in OREG, B in OREG, C in AOLIT |
| | POP A | A ← MEM(R7[RAW+USR+1:2]);<br>MHB ← MEM(R7[RAW+USR+1:2])[RDW-1:W]<br>R7 ← R7 – 4 | 2*1 | Pop from stack, A in OREG |
| | PUSH A | R7 ← R7 + 4<br>MEM(R7[RAW+USR+1:2])[W-1:0] ← A<br>MEM(R7[RAW+USR+1:2])[RDW-1:W] ← MHB | 2*1 | Push to stack, A in OREG |
| | MWRL A, C | MEM(C)[W-1:0] ← A | 3*2 | Memory Write Literal, A in OREG, C in ALIT |
| | MWRIL A, B | MEM(B[RAW+USR+1:2])[W-1:0] ← A | 3*2 | Memory Write Indirect Literal, A in OREG, B in OREG |
| ARU Transfer | ARD A, B,C | A ← ARU(C)[W-1:0]<br>B ← ARU(C)[2*W-1:W]<br>ACB ← ARU(C)[5+2*W:2*W] | ≥ 1 | Blocking ARU Read, A in AREG, B in AREG, C in ARDLIT |
| | AWR A, B, C | ARU(C)[W-1:0] ← A<br>ARU(C)[2*W:W] ← B<br>ARU(C)[5+2*W:2*W] ← ACB | ≥ 1 | Blocking ARU Write, A in OREG, B in OREG, C in AWRLIT |
| | ARDI A, B | A ← ARU(R6[8:0])[W-1:0]<br>B ← ARU(R6[8:0])[2*W-1:W]<br>ACB ← ARU(R6[8:0])[5+2*W:2*W] | ≥ 1 | Blocking ARU Read Indirect, A in AREG, B in AREG |
| | AWRI A, B | ARU(R6[4:0])[W-1:0] ← A<br>ARU(R6[4:0])[2*W-1:W] ← B<br>ARU(R6[4:0])[5+2*W:2*W] ← ACB | ≥ 1 | Blocking ARU Write Indirect, A in OREG, B in OREG |
| | NARD A, B, C | A ← ARU(C[8:0])[W-1:0]<br>B ← ARU(C[8:0])[2*W-1:W]<br>ACB ← ARU(C[8:0])[5+2*W:2*W] | ≥ 1*3 | Non-Blocking ARU Read, A in AREG, B in AREG |
| | NARDI A, B | A ← ARU(R6[8:0])[W-1:0]<br>B ← ARU(R6[8:0])[2*W-1:W]<br>ACB ← ARU(R6[8:0])[5+2*W:2*W] | ≥ 1*3 | Non-Blocking ARU Read Indirect, A in AREG, B in AREG |
| Arith. / Logic | ADDL A, C | A ← A + C | 1 | Add Literal, A in OREG, C in WLIT |
| | ADD A, B | A ← A + B | 1 | Add, A in OREG, B in OREG |
| | SUBL A, C | A ← A – C | 1 | Subtract Literal, A in OREG, C in WLIT |
| | SUB A, B | A ← A – B | 1 | Subtract, A in OREG, B in OREG |
| | NEG A, B | A ← – B | 1 | Negate, A in OREG, B in OREG |
| | ANDL A, C | A ← A AND C | 1 | AND Litral, A in OREG, C in WLIT |

Table 25.139 Instruction Set Summary (2/3)

| Class | Mnemonic | Operation | Instruction cycles | Synopsis |
|---|---|---|---|---|
| Arith. / Logic | AND A, B | A ⟵ A AND B | 1 | AND, A in OREG, B in OREG |
| | ORL A, C | A ⟵ A OR C | 1 | OR Literal, A in OREG, C in WLIT |
| | OR A, B | A ⟵ A OR B | 1 | OR, A in OREG, B in OREG |
| | XORL A, C | A ⟵ A XOR C | 1 | XOR Literal, A in OREG, C in WLIT |
| | XOR A, B | A ⟵ A XOR B | 1 | XOR, A in OREG, B in OREG |
| | SHR A, C | A ⟵ A >> C | 1 | Shift Right, A in OREG, C in SFTLIT |
| | SHL A, C | A ⟵ A << C | 1 | Shift Left, A in OREG, C in SFTLIT |
| | ASRU A, B | A ⟵ A >> B | 1 | Shift Right, A in OREG, B in OREG |
| | ASRS A, B | A ⟵ A >> B | 1 | Shift Right, A in OREG, B in OREG |
| | ASL A, B | A ⟵ A << B | 1 | Shift Left, A in OREG, B in OREG |
| | MINU A, B | A ⟵ MIN(A, B) | 1 | Minimum Unsigned, A in OREG, B in OREG |
| | MINS A, B | A ⟵ MIN(A, B) | 1 | Minimum Signed, A in OREG, B in OREG |
| | MAXU A, B | A ⟵ MAX(A, B) | 1 | Maximum Unsigned, A in OREG, B in OREG |
| | MAXS A, B | A ⟵ MAX(A, B) | 1 | Maximum Signed, A in OREG, B in OREG |
| Test | ATUL A, C | A < C ⇔ CY is set<br>A = C ⇔ Z is set | 1 | Arithmetic Test Unsigned Literal,<br>A in OREG, C in WLIT |
| | ATU A, B | A < B ⇔ CY is set<br>A = C ⇔ Z is set | 1 | Arithmetic Test Unsigned,<br>A in OREG, B in OREG |
| | ATSL A, C | A < C ⇔ CY is set<br>A = C ⇔ Z is set | 1 | Arithmetic Test Signed Literal,<br>A in OREG, C in WLIT |
| | ATS A, B | A < B ⇔ CY is set<br>A = C ⇔ Z is set | 1 | Arithmetic Test Signed,<br>A in OREG, B in OREG |
| | BTL A, C | A AND C | 1 | Bit Test Literal, A in OREG, C in WLIT |
| | BT A, B | A AND B | 1 | Bit Test, A in OREG, B in OREG |

**Table 25.139 Instruction Set Summary (3/3)**

| Class | Mnemonic | Operation | Instruction cycles | Synopsis |
|---|---|---|---|---|
| Control Flow | JMP C | PC ← C << 2 | 1*4 | Unconditional Jump, C in ALIT |
| | JBS A, B, C | PC ← C << 2 if A[B] is set | 1*5 | Jump if Bit Set, A in OREG, B in BITLIT |
| | JBC A, B, C | PC ← C << 2 if A[B] is clear | 1*5 | Jump if Bit Cleared, A in OREG, B in BITLIT |
| | CALL C | R7 ← R7 + 4<br>MEM(R7[RAW+USR+1:2]) ← PC+ 4<br>PC ← C << 2 | 2*6 | Call Subroutine, C in ALIT |
| | RET | PC ←<br>MEM(R7[RAW+USR+1:2])[RAW+USR+1:0]<br>R7 ← R7 - 4 | 2*6 | Return from Subroutine |
| Others | WURM A, B, C | wait until<br>A = (B AND ((0xFF << 16) + C)) | ≥ 1*7 | Wait Until Registers Match, A in OREG, B in OREG, C in MSKLIT |
| | NOP | | 1 | No Operation |

Note 1. Not faster than 1+NPS clock cycles due to pipeline flushing.
Note 2. Not faster than 1+2*NPS clock cycles due to pipeline flushing.
Note 3. Always faster than one ARU round trip cycle.
Note 4. Not faster than NPS clock cycles due to pipeline flushing.
Note 5. If the jump is executed, it is not faster than NPS clock cycles due to pipeline flushing.
Note 6. Not faster than 2*NPS clock cycles due to pipeline flushing.
Note 7. Fastest latency for resuming up from a suspended WURM instruction is 2+NPS clock cycles.

### 25.13.5.1 Instruction Codes

Table 25.140 Instruction Codes (1/2)

| Mnemonic | Instruction Code |
|---|---|
| MOVL | 0001aaaaccccccccccccccccccccccccc |
| MOV | 1010aaaabbbb0000---------------- |
| MRD | 1010aaaa----0001----cccccccccc-- |
| MWR | 1010aaaa----0010----cccccccccc-- |
| MRDI | 1010aaaabbbb0011----cccccccccc-- |
| MWRI | 1010aaaabbbb0100----cccccccccc-- |
| PUSH | 1010aaaa----0110---------------- |
| POP | 1010aaaa----0101---------------- |
| MWRL | 1010aaaa----0111----cccccccccc-- |
| MWRIL | 1010aaaabbbb1000---------------- |
| ARD | 1011aaaabbbb0000-------ccccccccc |
| AWR | 1011aaaabbbb0001----------ccccc |
| ARDI | 1011aaaabbbb0100---------------- |
| AWRI | 1011aaaabbbb0101---------------- |
| NARD | 1011aaaabbbb0010-------ccccccccc |
| NARDI | 1011aaaabbbb0011---------------- |
| ADDL | 0010aaaaccccccccccccccccccccccccc |
| ADD | 1100aaaabbbb0000---------------- |
| SUBL | 0011aaaaccccccccccccccccccccccccc |
| SUB | 1100aaaabbbb0001---------------- |
| NEG | 1100aaaabbbb0010---------------- |
| ANDL | 0100aaaaccccccccccccccccccccccccc |
| AND | 1100aaaabbbb0011---------------- |
| ORL | 0101aaaaccccccccccccccccccccccccc |
| OR | 1100aaaabbbb0100---------------- |
| XORL | 0110aaaaccccccccccccccccccccccccc |
| XOR | 1100aaaabbbb0101---------------- |
| SHR | 1100aaaa----0110----------ccccc |
| SHL | 1100aaaa----0111----------ccccc |
| ASL | 1101aaaabbbb0011---------------- |
| ASRU | 1101aaaabbbb0100---------------- |
| ASRS | 1101aaaabbbb0101---------------- |
| MINU | 1100aaaabbbb1100---------------- |
| MINS | 1100aaaabbbb1101---------------- |
| MAXU | 1100aaaabbbb1110---------------- |
| MAXS | 1100aaaabbbb1111---------------- |
| ATUL | 0111aaaaccccccccccccccccccccccccc |
| ATU | 1101aaaabbbb0000---------------- |
| ATSL | 1000aaaaccccccccccccccccccccccccc |
| ATS | 1101aaaabbbb0001---------------- |
| BTL | 1001aaaaccccccccccccccccccccccccc |
| BT | 1101aaaabbbb0010---------------- |

**Table 25.140 Instruction Codes (2/2)**

| Mnemonic | Instruction Code |
|---|---|
| JMP | 1110--------0000----cccccccccc-- |
| JBS | 1110aaaabbbb0001----cccccccccc-- |
| JBC | 1110aaaabbbb0010----cccccccccc-- |
| CALL | 1110--------0011----cccccccccc-- |
| RET | 1110--------0100---------------- |
| WURM | 1111aaaabbbb0000cccccccccccccccc |
| NOP | 0000---------------------------- |

The individual instructions are decoded by evaluating the bits ‘0’ and ‘1’ at its expected positions, as mentioned in the table above. If the instruction decoder detects an invalid combination of these bits, the corresponding MCS-channel is disabled and the ERR bit in the register STA is set. Bit positions marked as ‘–’ are not relevant for the instruction. The bit position ‘a’, ‘b’, and ‘c’ are reserved for binary encoding of the instruction arguments A, B, and C.

### 25.13.5.2 Data Transfer Instructions

#### (1) MOVL Instruction

Syntax: MOVL A, C

Operation: A ← C

Status: Z

Duration: 1 instruction cycle

Description: Transfer literal value C (C ∈ WLIT) to register A (A∈ OREG).
The zero bit Z of status register STA is set, if the transferred value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

#### (2) MOV Instruction

Syntax: MOV A, B

Operation: A ← B

Status: Z

Duration: 1 instruction cycle

Description: Transfer register B (B ∈ OREG) to register A (A ∈ OREG).
The zero bit Z of status register STA is set, if the transferred value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

#### (3) MRD Instruction

Syntax: MRD A, C

Operation: A ← MEM(C)[W-1:0];
MHB ← MEM(C)[RDW-1:W]

Status: Z

Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Transfer the lower W bits of memory content at location C (C ∈ ALIT) to register A (A ∈ OREG).
The upper RDW-W bits of the memory content at location C are transferred to the MHB register.
The zero bit Z of status register STA is set, if the lower W bits of the transferred value are zero, otherwise the zero bit is cleared.
If the MHB register is selected as destination register A (A ∈ OREG), the bits 0 to RDW-W-1 of the referred memory location are transferred to MHB.
The program counter PC is incremented by the value 4.

### (4) MWR Instruction

Syntax: MWR A, C

Operation: MEM(C)[W-1:0] ← A;
MEM(C)[RDW-1:W] ← MHB

Status: —

Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Transfer W bit value of register A (A ∈ OREG) together with the MHB register to the memory at location C (C ∈ ALIT).
The W bit value of register A is stored in the lower significant bits (bit 0 to W-1) of the memory location.
The MHB register is stored in bits W to RDW-W-1 of the referred memory location.
The program counter PC is incremented by the value 4.

### (5) MWRL Instruction

Syntax: MWRL A, C

Operation: MEM(C)[W-1:0] ← A

Status: —

Duration: 3 instruction cycles but not faster than 1+2*NPS clock cycles due to pipeline flushing.

Description: Transfer W bit value of register A (A ∈ OREG) to memory at location C (C ∈ ALIT).
The W bit value of register A is stored in the lower significant bits (bit 0 to W-1) of the memory location and the bits W to RDW-W are left unchanged.
The program counter PC is incremented by the value 4.
It should be noted that this operation is not an atomic instruction.

### (6) MRDI Instruction

Syntax: MRDI A, B [, C]

Operation: A ← MEM(B[RAW+USR+1:2] + C)[W-1:0]
MHB ← MEM(B[RAW+USR+1:2] + C)[RDW-1:W]

Status: Z

Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Transfer the bits 0 to W–1 of a memory location to register A (A ∈ OREG) using indirect addressing.
The upper RDW-W bits of this memory location are transferred to MHB register.
The memory location where to read from depends on register B (B ∈ OREG) and literal C (C ∈ AOLIT) and it is defined as B[RAW+USR+1:2] + C.
If the optional operand C is not available in the assembler syntax, the MCS assembler generates code with a default value of 0 for operand C.
The zero bit Z of status register STA is set, if the transferred bits 0 to W-1 are zero, otherwise the zero bit is cleared.
If the MHB register is selected as destination register A (A ∈ OREG), the bits 0 to RDW–W–1 of the referred memory location are transferred to MHB.
The program counter PC is incremented by the value 4.

### (7) MWRI Instruction

Syntax: MWRI A, B [, C]

Operation: MEM(B[RAW+USR+1:2] + C)[W-1:0] ← A;
MEM(B[RAW+USR+1:2] + C)[RDW-1:W] ← MHB

Status: —

Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Transfer value of register A (A ∈ OREG) to the least significant bits 0 to W–1 of a memory location using indirect addressing.
The MHB register is moved to the bits W to RDW-1 at the same memory location.
The memory location where to write to depends on register B (B ∈ OREG) and literal C (C ∈ AOLIT) and it is defined as B[RAW+USR+1:2] + C.
If the optional operand C is not available in the assembler syntax, the MCS assembler generates code with a default value of 0 for operand C.
The program counter PC is incremented by the value 4.

### (8) MWRIL Instruction

Syntax: MWRIL A, B

Operation: MEM(B[RAW+USR+1:0])[W-1:0] ← A;

Status: —

Duration: 3 instruction cycles but not faster than 1+2*NPS clock cycles due to pipeline flushing.

Description: Transfer W bit value of A (A ∈ OREG) to memory using indirect addressing.
The memory location where to write to is defined by the bits 2 to RAW+1 of register B (B ∈ OREG).
The W bit value is stored in the lower significant bits (bit 0 to W-1) of the memory location and the bits W to RDW–1 are left unchanged.
The program counter PC is incremented by the value 4.
It should be noted that this operation is not an atomic instruction.

### (9) POP Instruction

Syntax: POP A

Operation: A ← MEM(R7[RAW+USR+1:2])[W-1:0];
R7 ← R7 – 4;
MHB ← MEM(R7[RAW+USR+1:2])[RDW–1:W];
SP_CNT ← SP_CNT – 1

Status: Z, EN

Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Transfer the lower significant bits (bit 0 to W–1) from the top of stack to register A (A ∈ OREG), followed by decrementing the stack pointer register R7 with the value 4.
The upper bits W to RDW-1 from the top of the stack are transferred to register MHB.
If the MHB register is selected as destination register A (A ∈ OREG), the bits 0 to RDW–W–1 from the top of the stack are transferred to MHB.
The memory location for the top of the stack is identified by the bits 2 to RAW+1 of the stack pointer register R7.
The zero bit Z of status register STA is set, if the lower W bit of the transferred value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.
The SP_CNT bit field inside the GTM0MCSixCTRL register is decremented.
If an underflow on the SP_CNT bit field occurs, the STK_ERR[i]_IRQ is raised.
If an underflow on the SP_CNT bit field occurs and the bit HLT_SP_OFL of register MCS[i]_CTRL is set, the current MCS-channel is disabled by clearing the EN bit of STA.

#### (10) PUSH Instruction

Syntax: PUSH A

Operation:
R7 ← R7 + 4;
MEM(R7[RAW+USR+1:2])[W-1:0] ← A;
MEM(R7[RAW+USR+1:2])[RDW-1:W] ← MHB
SP_CNT ← SP_CNT + 1;

Status: EN

Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Increment the stack pointer register R7 with the value 4, followed by transferring a W bit value of operand A (A ∈ OREG) together with a MHB register to the new top of the stack. The W bit value of A is stored in the bits 0 to W-1 of the memory location.
The content of the MHB register is stored in the bit W to RDW-1 of the memory location.
The memory location for the top of the stack is referred by the bits 2 to RAW+1 of the stack pointer register.
The program counter PC is incremented by the value 4.
The SP_CNT bit field inside the GTM0MCSixCTRL register is incremented.
If an overflow on the SP_CNT bit field occurs, the STK_ERR[i]_IRQ is raised.
If an overflow on the SP_CNT bit field occurs and the bit HLT_SP_OFL of register MCS[i]_CTRL is set, the current MCS-channel is disabled by clearing the EN bit of STA.
If an overflow on the SP_CNT bit field occurs and the bit HLT_SP_OFL of register MCS[i]_CTRL is set, the memory write operation for the A and MHB is discard.

### 25.13.5.3 ARU Instructions

#### (1) ARD Instruction

Syntax: ARD A, B, C

Operation:
A ← ARU(C)[W-1:0];
B ← ARU(C)[2*W-1:W];
ACB ← ARU(C)[4+2*W:2*W]

Status: CAT

Duration: suspends current MCS-channel

Description: Perform a blocking read access to the ARU and transfer both W bit values received at the ARU port to the registers A and B (A ∈ AREG, B ∈ AREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.
If A and B see the same register, only the upper W bit ARU word is stored and the lower W bit ARU word is discard.
If any transferred W bit value from the ARU should not stored in a register, the dummy register ZERO ∈ AREG can be selected in A or B to discard the corresponding ARU

data. Actually, all address values of A and B that exceed the range 0 to 7 discard the corresponding ARU data.
The received ARU control bits are stored in the register ACB.
The literal C (C ∈ ARDLIT) define the ARU address where to read from.
At the beginning of the instruction execution the CAT bit in the register STA is always cleared. After the execution of the instruction the CAT bit is updated in order to show if the instruction finished successfully (CAT = 0) or it was cancelled by the CPU (CAT = 1).
The program counter PC is incremented by the value 4.

### (2) ARDI Instruction

Syntax: ARDI A, B

Operation:
A ← ARU(R6[8:0])[W-1:0];
B ← ARU(R6[8:0])[2*W-1:W];
ACB ← ARU(R6[8:0])[4+2*W:2*W]

Status: CAT

Duration: suspends current MCS-channel

Description: Perform a blocking read access to the ARU and transfer both W bit values received at the ARU port to the registers A and B (A ∈ AREG, B ∈ AREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.
If A and B see the same register, only the upper W bit ARU word is stored and the lower W bit ARU word is discard.
If any transferred W bit value from the ARU should not stored in a register, the dummy register ZERO ∈ AREG can be selected in A or B to discard the corresponding ARU data. Actually, all address values of A and B that exceed the range 0 to 7 discard the corresponding ARU data.
The received ARU control bits are stored in the register ACB.
The read address is obtained from the bits 0 to 8 of the channels register R6.
At the beginning of the instruction execution the CAT bit in the register STA is always cleared. After the execution of the instruction the CAT bit is updated in order to show if the instruction finished successfully (CAT = 0) or it was cancelled by the CPU (CAT = 1).
The program counter PC is incremented by the value 4.

### (3) AWR Instruction

Syntax: AWR A, B, C

Operation:
ARU(C)[W-1:0] ← A;
ARU(C)[2*W-1:W] ← B;
ARU(C)[4+2*W:2*W] ← ACB;

Status: CAT

Duration: suspends current MCS-channel

Description: Perform a blocking write access to the ARU and transfer two W bit values to the ARU port using the registers A and B (A ∈ OREG, B ∈ OREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.
The ARU control bits to be sent are taken from the register ACB.
The literal C (C ∈ AWRLIT) define an index into the pool of ARU write address that is used for writing data.

Each MCS sub module has a pool of several write addresses that can be shared between all MCS-channels arbitrarily.
At the beginning of the instruction execution the CAT bit in the register STA is always cleared. After the execution of the instruction the CAT bit is updated in order to show if the instruction finished successfully (CAT = 0) or it was cancelled by the CPU (CAT = 1).
The program counter PC is incremented by the value 4.

### (4) AWRI Instruction

**Syntax**: AWRI A, B

Operation: ARU(R6[4:0])[W-1:0] ← A;
ARU(R6[4:0])[2*W-1:W] ← B;
ARU(R6[4:0])[4+2*W:2*W] ← ACB;

**Status**: CAT

**Duration**: suspends current MCS-channel

Description: Perform a blocking write access to the ARU and transfer two W bit values to the ARU port using the registers A and B (A ∈ OREG, B ∈ OREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.
The ARU control bits to be sent are taken from the register ACB.
The bits 0 to 4 of the register R6 define an index into the pool of ARU write address that is used for writing data.
Each MCS sub module has a pool of several write addresses that can be shared between all MCS-channels arbitrarily.
At the beginning of the instruction execution the CAT bit in the register STA is always cleared. After the execution of the instruction the CAT bit is updated in order to show if the instruction finished successfully (CAT = 0) or it was cancelled by the CPU (CAT = 1).
The program counter PC is incremented by the value 4.

### (5) NARD Instruction

Syntax: NARD A, B, C

Operation: A ← ARU(C)[W-1:0];
B ← ARU(C)[2*W:W];
ACB ← ARU(C)[4+2*W:2*W]

Status: SAT

Duration: suspends current MCS-channel for a maximum of one ARU round trip cycle

Description: Perform a non-blocking read access to the ARU trying to transfer both W bit values received at the ARU port to the registers A and B (A ∈ AREG, B ∈ AREG), whereas A holds the lower W bit ARU word, B holds the upper W bit ARU word, and the ACB register holds the received ARU control bits.
The literal C (C ∈ ARDLIT) define the ARU address where to read from.
If the transfer finished successfully, the bit SAT of the register STA is set and the transferred values are stored in the registers A, B, and ACB.
If the transfer failed due to missing data at requested source, the bit SAT of the register STA is cleared and registers A, B, and ACB are not changed.
If A and B see the same register, only the upper W bit ARU word is stored and the lower W bit ARU word is discard.

If any transferred W bit value from the ARU should not stored in a register, the dummy register ZERO ∈ AREG can be selected in A or B to discard the corresponding ARU data. Actually, all address values of A and B that exceed the range 0 to 7 discard the corresponding ARU data.
The program counter PC is incremented by the value 4.

#### (6) NARDI Instruction

Syntax: NARDI A, B

Operation: A ← ARU(R6[8:0])[W-1:0];
B ← ARU(R6[8:0])[2*W-1:W];
ACB ← ARU(R6[8:0])[4+2*W:2*W]

Status: SAT

Duration: suspends current MCS-channel for a maximum of one ARU round trip cycle

Description: Perform a non-blocking read access to the ARU trying to transfer both W bit values received at the ARU port to the registers A and B (A ∈ AREG, B ∈ AREG), whereas A holds the lower W bit ARU word, B holds the upper W bit ARU word, and the ACB register holds the received ARU control bits.
The read address is obtained from the bits 0 to 8 of the channels register R6.
If the transfer finished successfully, the bit SAT of the register STA is set and the transferred values are stored in the registers A, B, and ACB.
If the transfer failed due to missing data at requested source, the bit SAT of the register STA is cleared and registers A, B, and ACB are not changed.
If A and B see the same register, only the upper W bit ARU word is stored and the lower 24 bit ARU word is discard.
If any transferred W bit value from the ARU should not stored in a register, the dummy register ZERO ∈ AREG can be selected in A or B to discard the corresponding ARU data. Actually, all address values of A and B that exceed the range 0 to 7 discard the corresponding ARU data.
The program counter PC is incremented by the value 4.

### 25.13.5.4 Arithmetic Logic Instructions

#### (1) ADDL Instruction

Syntax: ADDL A, C

Operation: A ← A + C

Status: Z, CY, N, V

Duration: 1 instruction cycle

Description: Perform addition operation of a register A (A ∈ OREG) with a W bit literal value C (C ∈ WLIT) and store the result in register A.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The carry bit CY of status register STA is set, if an unsigned overflow/underflow occurred during addition, otherwise the bit is cleared. An unsigned overflow has occurred when the result of the operation cannot be represented in the interval [0; $2^{W-1}$-1], assuming that both operands A and C are unsigned values within the interval [0; $2^{W-1}$-1].
The overflow bit V of status register STA is set, if a signed overflow/underflow

occurred during addition, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [$-2^{W-1}$; $2^{W-1}-1$], assuming that both operands A and C are signed values within the interval [$-2^{W-1}$; $2^{W-1}-1$].
The negative bit N of status register STA equals the most significant bit of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.
The program counter PC is incremented by the value 4.

### (2) ADD Instruction

Syntax: ADD A, B

Operation: A ← A + B

Status: Z, CY, N, V

Duration: 1 instruction cycle

Description: Perform addition operation of a register A (A ∈ OREG) with an operand B (B ∈ OREG) and store the result in register A.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The carry bit CY of status register STA is set, if an unsigned overflow occurred during addition, otherwise the bit is cleared. An unsigned overflow has occurred when the result of the operation cannot be represented in the interval [0; $2^{W-1}-1$], assuming that both operands A and B are unsigned values within the interval [0; $2^{W-1}-1$].
The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during addition, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [$-2^{W-1}$; $2^{W-1}-1$], assuming that both operands A and B are signed values within the interval [$-2^{W-1}$; $2^{W-1}-1$].
The negative bit N of status register STA equals the most significant bit of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.
The program counter PC is incremented by the value 4.

### (3) SUBL Instruction

Syntax: SUBL A, C

Operation: A ← A - C

Status: Z, CY, N, V

Duration: 1 instruction cycle

Description: Perform subtraction operation of a register A (A ∈ OREG) with a W bit literal value C (C ∈ WLIT) and store the result in register A.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The carry bit CY of status register STA is set, if an unsigned underflow occurred during subtraction, otherwise the bit is cleared. An unsigned underflow has occurred when the result of the operation cannot be represented in the interval [0; $2^{W-1}-1$], assuming that both operands A and C are unsigned values within the interval [0; $2^{W-1}-1$].
The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during subtraction, otherwise the bit is cleared. A signed overflow/underflow

has occurred when the result of the operation cannot be represented in the interval [$-2^{W-1}$; $2^{W-1}-1$], assuming that both operands A and C are signed values within the interval [$-2^{W-1}$; $2^{W-1}-1$].
The negative bit N of status register STA equals the most significant bit of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.
The program counter PC is incremented by the value 4.

### (4) SUB Instruction

Syntax: SUB A, B

Operation: A ← A – B

Status: Z, CY, N, V

Duration: 1 instruction cycle

Description: Perform subtraction operation of a register A (A ∈ OREG) with an operand B (B ∈ OREG) and store the result in register A.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The carry bit CY of status register STA is set, if an unsigned underflow occurred during subtraction, otherwise the bit is cleared. An unsigned underflow has occurred when the result of the operation cannot be represented in the interval [0; $2^{W-1}-1$], assuming that both operands A and B are unsigned values within the interval [0; $2^{W-1}-1$].
The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during subtraction, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [$-2^{W-1}$; $2^{W-1}-1$], assuming that both operands A and B are signed values within the interval [$-2^{W-1}$; $2^{W-1}-1$].
The negative bit N of status register STA equals the most significant bit of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.
The program counter PC is incremented by the value 4.

### (5) NEG Instruction

Syntax: NEG A, B

Operation: A ← –B

Status: Z, N, V

Duration: 1 instruction cycle

Description: Perform negation operation (2's Complement) with an operand B (B ∈ OREG) and store the result in a register A (A ∈ OREG).
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during subtraction, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [$-2^{W-1}$; $2^{W-1}-1$], assuming that both operands A and B are signed values within the interval [$-2^{W-1}$; $2^{W-1}-1$].
The negative bit N of status register STA equals the most significant bit of the operation result, in order to determine if a calculated signed result is negative (N = 1) or positive

(N = 0), assuming that no overflow/underflow occurred.
The program counter PC is incremented by the value 4.

### (6) ANDL Instruction

Syntax: ANDL A, C

Operation: A ← A AND C

Status: Z

Duration: 1 instruction cycle

Description: Perform bitwise AND conjunction of a register A (A ∈ OREG) with a W bit literal value C (C ∈ WLIT) and store the result in register A.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

### (7) AND Instruction

Syntax: AND A, B

Operation: A ← A AND B

Status: Z

Duration: 1 instruction cycle

Description: Perform bitwise AND conjunction of a register A (A ∈ OREG) with an operand B (B ∈ OREG) and store the result in register A.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

### (8) ORL Instruction

Syntax: ORL A, C

Operation: A ← A OR C

Status: Z

Duration: 1 instruction cycle

Description: Perform bitwise OR conjunction of a register A (A ∈ OREG) with a W bit literal value C (C ∈ WLIT) and store the result in register A.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

### (9) OR Instruction

Syntax: OR A, B

Operation: A ← A OR B

Status: Z

Duration: 1 instruction cycle

Description: Perform bitwise OR conjunction of a register A (A ∈ OREG) with an operand B (B ∈ OREG) and store the result in register A.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the

zero bit is cleared.
The program counter PC is incremented by the value 4.

### (10) XORL Instruction

Syntax: XORL A, C

Operation: A ← A XOR C

Status: Z

Duration: 1 instruction cycle

Description: Perform bitwise XOR conjunction of a register A (A ∈ OREG) with a W bit literal value C (C ∈ WLIT) and store the result in register A.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

### (11) XOR Instruction

Syntax: XOR A, B

Operation: A ← A XOR B

Status: Z

Duration: 1 instruction cycle

Description: Perform bitwise XOR conjunction of a register A (A ∈ OREG) with an operand B (B ∈ OREG) and store the result in register A.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

### (12) SHR Instruction

Syntax: SHR A, C

Operation: A ← A >> C

Status: Z, CY

Duration: 1 instruction cycle

Description: Perform right shift operation C (C ∈ SFTLIT) times of register A (A ∈ OREG). The most significant bits that are shifted into A are cleared.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The carry bit CY of status register STA is updated to the last LSB that is shifted out of the register.
The program counter PC is incremented by the value 4.

### (13) SHL Instruction

Syntax: SHL A, C

Operation: A ← A << C

Status: Z, CY

Duration: 1 instruction cycle

Description: Perform left shift operation C (C ∈ SFTLIT) times of register A (A ∈ OREG). The lower significant bits that are shifted into A are cleared.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The carry bit CY of status register STA is updated to the previous MSB that is shifted out of the register. It should be noticed, if the register A contains less than W bits, CY is always 0.
The program counter PC is incremented by the value 4.

### (14) ASRU Instruction

Syntax: ASRU A, B

Operation: A ← A >> B

Status: Z

Duration: 1 instruction cycle

Description: Perform arithmetic unsigned right shift operation, which means that the unsigned operand of register A (A ∈ OREG) is right shifted B times (B ∈ OREG). Operand B is also an unsigned type. The most significant bits that are shifted into A are cleared.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

### (15) ASRS Instruction

Syntax: ASRS A, B

Operation: A ← A >> B

Status: Z

Duration: 1 instruction cycle

Description: Perform arithmetic signed right shift operation, which means that the signed operand of register A (A ∈ OREG) is right shifted B times (B ∈ OREG). Operand B is an unsigned type. The operation also performs a sign extension, which means that value of the most significant bits that are shifted into A are determined by the most significant bit of the original operand A.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

### (16) ASL Instruction

Syntax: ASL A, B

Operation: A ← A << B

Status: Z, CY, V

Duration: 1 instruction cycle

Description: Perform arithmetic left shift operation for signed and unsigned numbers, which means that the operand of register A (A ∈ OREG) is left shifted B times (B ∈ OREG). Operand B is always an unsigned type.
The carry bit CY of status register STA is set, if an unsigned overflow occurred during shifting, otherwise the bit is cleared. An unsigned overflow has occurred if the

calculated result $A*2^B$ cannot be represented in the interval [0; $2^W$-1], assuming that both operands A and B are unsigned values within the interval [0; $2^W$-1].
The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during shifting, otherwise the bit is cleared. A signed overflow/underflow has occurred when the calculated result $A*2^B$ cannot be represented in the interval [$-2^{W-1}$; $2^{W-1}$-1], assuming that signed operand A is within the interval [$-2^{W-1}$; $2^{W-1}$-1] and the unsigned operand B is within the interval [0; $2^{W-1}$-1].
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

### (17) MINU Instruction

Syntax: MINU A, B

Operation: A ← MIN(A, B)

Status: Z

Duration: 1 instruction cycle

Description: Determine the minimum of an unsigned operand A (A ∈ OREG) and an unsigned operand B (B ∈ OREG). If A is less than or equal to B, A is left unchanged. Otherwise, if A is greater than B, the operand B is moved to A.
The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

### (18) MINS Instruction

Syntax: MINS A, B

Operation: A ← MIN(A, B)

Status: Z

Duration: 1 instruction cycle

Description: Determine the minimum of a signed operand A (A ∈ OREG) and a signed operand B (B ∈ OREG). If A is less than or equal to B, A is left unchanged. Otherwise, if A is greater than B, the operand B is moved to A.
The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

### (19) MAXU Instruction

Syntax: MAXU A, B

Operation: A ← MAX(A, B)

Status: Z

Duration: 1 instruction cycle

Description: Determine the maximum of an unsigned operand A (A ∈ OREG) and an unsigned operand B (B ∈ OREG). If A is greater than or equal to B, A is left unchanged. Otherwise, if A is less than B, the operand B is moved to A.
The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

#### (20) MAXS Instruction

Syntax: MAXS A, B

Operation: A ← MAX(A, B)

Status: Z

Duration: 1 instruction cycle

Description: Determine the maximum of a signed operand A (A ∈ OREG) and a signed operand B (B ∈ OREG). If A is greater than or equal to B, A is left unchanged. Otherwise, if A is less than B, the operand B is moved to A.
The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

### 25.13.5.5 Test Instructions

#### (1) ATUL Instruction

Syntax: ATUL A, C

Operation: A – C

Status: Z, CY

Duration: 1 instruction cycle

Description: Arithmetic Test with an unsigned operand A (A ∈ OREG) and an unsigned W bit literal value C (C ∈ WLIT).
The carry bit CY of status register STA is set if unsigned operand A is less than unsigned literal C.
Otherwise, the carry bit CY of status register STA is cleared if unsigned operand A is greater than or equal to unsigned literal C.
The zero bit Z of status register STA is set, if A equals to C.
Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to C.
The program counter PC is incremented by the value 4.

#### (2) ATU Instruction

Syntax: ATU A, B

Operation: A – B

Status: Z, CY

Duration: 1 instruction cycle

Description: Arithmetic Test with an unsigned operand A (A ∈ OREG) and an unsigned operand B (B ∈ OREG).
The carry bit CY of status register STA is set if unsigned operand A is less than unsigned operand B.
Otherwise, the carry bit CY of status register STA is cleared if unsigned operand A is greater than or equal to unsigned operand B.
The zero bit Z of status register STA is set, if A equals to B.
Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to B. The program counter PC is incremented by the value 4.

### (3) ATSL Instruction

Syntax: ATSL A, C

Operation: A - C

Status: Z, CY

Duration: 1 instruction cycle

Description: Arithmetic Test with a signed operand A (A ∈ OREG) and a signed W bit literal value C (C ∈ WLIT).
The carry bit CY of status register STA is set if signed operand A is less than signed literal C.
Otherwise, the carry bit CY of status register STA is cleared if signed operand A is greater than or equal to signed literal C.
The zero bit Z of status register STA is set, if A equals to C.
Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to C. The program counter PC is incremented by the value 4.

### (4) ATS Instruction

Syntax: ATS A, B

Operation: A – B

Status: Z, CY

Duration: 1 instruction cycle

Description: Arithmetic Test with a signed operand A (A ∈ OREG) and a signed operand B (B ∈ OREG).
The carry bit CY of status register STA is set if signed operand A is less than signed operand B.
Otherwise, the carry bit CY of status register STA is cleared if signed operand A is greater than or equal to signed operand B.
The zero bit Z of status register STA is set, if A equals to B.
Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to B. The program counter PC is incremented by the value 4.

### (5) BTL Instruction

Syntax: BTL A, C

Operation: A AND C

Status: Z

Duration: 1 instruction cycle

Description: Bit test of an operand A (A ∈ OREG) with a W bit literal bit mask C (C ∈ WLIT).
The bit test is performed by applying a bitwise logical AND operation with operand A and the bit mask C without storing the result.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

#### (6) BT Instruction

Syntax: BT A, B

Operation: A AND B

Status: Z

Duration: 1 instruction cycle

Description: Bit test of an operand A (A ∈ OREG) with an operand B (B ∈ OREG), whereas usually one of the operands is a register holding a bit mask.
The bit test is performed by applying a bitwise logical AND operation with register A and register B without storing the result.
The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.
The program counter PC is incremented by the value 4.

### 25.13.5.6 Control Flow Instructions

#### (1) JMP Instruction

Syntax: JMP C

Operation: PC ← C << 2

Status: —

Duration: 1 instruction cycle but not faster than NPS clock cycles due to pipeline flushing.

Description: Execute unconditional jump to the memory location C (C ∈ ALIT).
The program counter PC is loaded with literal C.

#### (2) JBS Instruction

Syntax: JBS A, B, C

Operation: PC ← C << 2 if A[B] is set

Status: —

Duration: 1 instruction cycle but if the jump is executed, it is not faster than NPS clock cycles due to pipeline flushing.

Description: Execute conditional jump to the memory location C (C ∈ ALIT).
The program counter PC is loaded with literal C, if the bit at position B (B ∈ BITLIT) of operand A (A ∈ OREG) is set.
Otherwise, if the bit is cleared, the program counter PC is incremented by the value 4.

#### (3) JBC Instruction

Syntax: JBC A, B, C

Operation: PC ← C << 2 if A[B] is cleared

Status: —

Duration: 1 instruction cycle but if the jump is executed, it is not faster than NPS clock cycles due to pipeline flushing.

Description: Execute conditional jump to the memory location C (C ∈ ALIT).
The program counter PC is loaded with literal C, if the bit at position B (B ∈ BITIT) of operand A (A ∈ OREG) is cleared.
Otherwise, if the bit is set, the program counter PC is incremented by the value 4.

### (4) CALL Instruction

Syntax: CALL C

Operation:
R7 ← R7 + 4;
MEM(R7[RAW+USR+1:2])[RAW+USR+1:2] ← PC + 4;
PC ← C << 2;
SP_CNT ← SP_CNT + 1

Status: EN

Duration: 2 instruction cycles but not faster than 2*NPS clock cycles due to pipeline flushing.

Description: Call subprogram at memory location C (C ∈ ALIT).
The stack pointer register R7 is incremented by the value 4.
The memory location for the top of the stack is identified by the bits 2 to RAW+1 of the stack pointer register.
After the stack pointer is incremented, the incremented value of the PC is transferred to the top of the stack.
The program counter PC is loaded with literal C.
The SP_CNT bit field inside the GTM0MCSixCTRL register is incremented. If an overflow on the SP_CNT bit field occurs, the STK_ERR[i]_IRQ is raised.
If an overflow on the SP_CNT bit field occurs and the bit HLT_SP_OFL of register GTM0MCSiCTRLSTAT is set, the channel current MCS-channel is disabled by clearing the EN bit of STA.
If an overflow on the SP_CNT bit field occurs and the bit HLT_SP_OFL of register GTM0MCSiCTRLSTAT is set, the memory write operation of the incremented PC is discarding.

### (5) RET Instruction

Syntax: RET

Operation:
PC ← MEM(R7[RAW+USR+1:2])[RAW+USR+1:0];
R7 ← R7 - 4;
SP_CNT ← SP_CNT – 1

Status: EN

Duration: 2 instruction cycles but not faster than 2*NPS clock cycles due to pipeline flushing.

Description: Return from subprogram.
The program counter PC is loaded with current value on the top of the stack.
Finally, the stack pointer register R7 is decremented by the value 4.
The memory location for the top of the stack is identified by the bits 2 to RAW+1 of the stack pointer register.
The SP_CNT bit field inside the GTM0MCSixCTRL register is decremented.
If an underflow on the SP_CNT bit field occurs, the STK_ERR[i]_IRQ is raised.
If an underflow on the SP_CNT bit field occurs and the bit HLT_SP_OFL of register MCS[i]_CTRL is set, the channel current MCS-channel is disabled by clearing the EN bit of STA.

### 25.13.5.7 Other Instructions

#### (1) WURM Instruction

Syntax: WURM A, B, C

Operation: Wait until register match.

Status: CWT

Duration: Suspends current MCS-channel. Fastest latency for resuming up from a suspended WURM instruction is 2+NPS clock cycles.

Description: Suspend current MCS-channel until the following register match condition occurs:

A = (B AND MASK),
whereas A ∈ OREG, B ∈ OREG, AND is a bitwise AND operation with bitmask MASK. The bits 16 to 23 of MASK are set to true and the bits 0 to 15 are copied from the instructions literal C ∈ MSKLIT. If the match condition evaluates to true, the suspended MCS channel is resumed and the program counter PC is incremented by the value 4 meaning that the MCS channel continues its program. However, if the match condition is true at the beginning of the instruction execution, the instruction does not suspend the channel and the program counter PC is incremented by the value 4.
At the beginning of the instruction execution the CWT bit in the register STA is always cleared. After the execution of the instruction the CWT bit is updated in order to show if the instruction finished successfully (CWT = 0) or it was cancelled by the CPU (CWT = 1). If the CWT bit is set simultaneously with the occurrence of the register match condition, the register match condition has the higher priority resulting in a cleared CWT bit.
This instruction can be used to wait for one or more trigger events generated by other MCS-channels or the CPU. In this case register B is the trigger register STRG, A is a general purpose register holding the bits with the trigger condition to wait for and C is the bitmask that enables trigger bits of interest. The trigger bits can be set by other MCS channels with a write access (e.g. using a MOVL instruction) to the STRG register or the CPU with a write access to the GTM0MCSiSTRG register. The trigger bits are not cleared automatically by hardware after resuming an MCS-channel, but they have to be cleared explicitly with a write access to the register CTRG by the MCS-channel or with a write access to the register GTM0MCSiCTRG by the CPU. Please note that more than one channel can wait for the same trigger bit to continue.
The instruction can also be used to wait on a specific time/angle event provided by the TBU. In this case register B is the interesting TBU register TBU_TS0 or TBU_TS1, register A is a general purpose register holding the value to wait for and bitmask C should be set to 0xFFFF.

#### (2) WURMX Instruction

Syntax: WURMX A, B

#### (3) NOP Instruction

Syntax: NOP

Operation: —

Status: —

Duration: 1 instruction cycle

Description: No operation is performed.
The program counter PC is incremented by the value 4.

## 25.13.6 MCS Internal Register Overview

### 25.13.6.1 MCS Internal Registers Overview

**Table 25.141 Register list**

| Symbol | Register Name | Details in Section |
|---|---|---|
| R[x] | General Purpose Register x (x = 0 to 7), | **25.13.7.1** |
| STA | Status Register | **25.13.7.2** |
| ACB | ARU Control Bit Register | **25.13.7.3** |
| CTRG | Clear Trigger Bits Register | **25.13.7.4** |
| STRG | Set Trigger Bits Register | **25.13.7.5** |
| TBU_TS0 | TBU Timestamp TS0 Register | **25.13.7.6** |
| TBU_TS1 | TBU Timestamp TS1 Register | **25.13.7.7** |
| MHB | Memory High Byte Register | **25.13.7.8** |

## 25.13.7 MCS Internal Register Description

This section describes the MCS internal registers that can be directly addressed with the MCS instruction set. Many of the registers can also be addressed by the CPU but with another Register Label (for details see **Section 25.13.9, MCS Configuration Register Description** ). Some of the internal registers are also shared between neighboring MCS channels.

### 25.13.7.1 Register R[y] (y = 0 to 7)

**Access:** This register can be read/written in 24-bit units.

**Address:** —

**Value after reset:** $00\ 0000_H$

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.142 R[y] Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 23 to 0 | DATA | Data field of general purpose register. |

**NOTES**

1. Register R4 is a commonly used register, which can be accessed by all MCS-channels and the CPU. This register can be used for fast data transfer between the MCS-channels.
2. Register R5 is a commonly used register, which can be accessed by all MCS-channels and the CPU. This register can be used for fast data transfer between the MCS-channels.
3. Register R6 used also as index/address register for indirect ARU addressing instructions.
4. Register R7 is also used as stack pointer register, if stack operations are used in the MCS micro program.

### 25.13.7.2 Register STA

**Access:** This register can be read in 24-bit units.

**Address:** —

**Value after reset:** $00\ 0000_H$

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SP_CNT | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | — | — | — | — | — | SAT | CWT | CAT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | N | V | Z | CY | MCA | ERR | IRQ | EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 25.143 STA Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 23 to 19 | Reserved | These bits are always read as 0. |
| 18 to 16 | SP_CNT | Stack pointer counter value.<br>**NOTE**<br>Actual stack depth of channel. The bit field is incremented on behalf of a CALL or PUSH instruction and decremented on behalf of a RET or POP instruction. The MCS channel STK_ERR_IRQ is raised, when an overflow or underflow is detected on this bit field. |
| 15 to 11 | Reserved | These bits are always read as 0. |
| 10 | SAT | Successful ARU transfer bit.<br>0: Non-blocking ARU transfer failed due to missing data.<br>1: Non-blocking ARU transfer finished successfully. |
| 9 | CWT | Cancel WURM instruction bit.<br>0: Last WURM instruction was not cancelled.<br>1: CPU cancelled last WURM instruction of channel.<br>**NOTE**<br>This bit is updated after each WURM instruction and it should be evaluated immediately after the WURM instruction. Otherwise, the CPU could set the bit leading to a bad status information in the MCS program. |
| 8 | CAT | Cancel ARU transfer bit.<br>0: Last ARU transfer was not cancelled.<br>1: CPU cancelled last ARU transfer.<br>**NOTE**<br>This bit is updated after each ARU transfer and it should be evaluated immediately after the ARU instruction. Otherwise, the CPU could set the bit leading to a bad status information in the MCS program. |

**Table 25.143 STA Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | N | Negative bit.<br>The negative bit is updated by arithmetic instructions in order to indicate a negative result. |
| 6 | V | Overflow bit.<br>The overflow bit is updated by arithmetic instructions in order to indicate a signed under/overflow. |
| 5 | Z | Zero bit.<br>The zero bit is updated by several arithmetic, logic and data transfer instructions to indicate a result of zero. |
| 4 | CY | Carry bit.<br>The carry bit is updated by several arithmetic and logic instructions. In arithmetic operations, the carry bit indicates an unsigned under/overflow. |
| 3 | MCA | MON Activity signalling for MCS channel.<br>0: No activity signalled to sub module MON.<br>1: Activity signalled to sub module MON.<br>**NOTE**<br>When this bit is set the corresponding channel in the MON sub module register GTMMONACTIVITY0 is set (see **Section 25.17.8.2, GTMMONACTIVITY0**. This bit is automatically cleared after writing it by the MCS channel program. |
| 2 | ERR | Set Error Signal.<br>0: No Error occurred.<br>1: Error occurred.<br>**NOTES**<br>1. The ERR bit of an MCS-channel reflects an Error status that may be caused by one of the following conditions:<br>– MCS-channel sets the ERR bit by software (e.g. with instruction ORL STA, 0x4)<br>– ECC RAM Error occurred while accessing the connected RAM (also disables the MCS-channel by clearing bit EN and the first error occurred updates bit field ERR_SRC_ID of register GTM0MCSiCTRLSTAT)<br>– Decoding an instruction with an invalid opcode (also disables the MCS-channel by clearing bit EN and the first error occurred updates bit field ERR_SRC_ID of register GTM0MCSiCTRLSTAT)<br>– A memory address range overflow occurred (also disables the MCS-channel by clearing bit EN and the first error occurred updates bit field ERR_SRC_ID of register GTM0MCSiCTRLSTAT)<br>2. If the ERR bit is set due to a memory address range overflow any read or write access to the RAM is blocked.<br>3. If the GTM includes a MON sub module, the ERR signal is always captured by this module.<br>4. An MCS-channel can set the error bit by writing value 1 to bit ERR. Writing a value 0 to this bit does not cancel the error signal, and thus has no effect. In Addition, writing a value 1 to ERR always triggers the ERR interrupt, independently from the current state of the error signal.<br>5. The ERR bit can only be cleared by CPU, by writing a 1 to the GTM0MCSiERR register (see **Section 25.13.9.17, GTM0MCSiERR (i = 0, 1)** ).<br>6. An MCS-channel can read the ERR bit in order to determine the current state of the error signal. The MCS-channel reads a value 1 if an ERR occurred previously, but not cleared by CPU. If an MCS-channel reads a value 0 no error was set or it has been cleared by CPU. |

**Table 25.143 STA Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | IRQ | Trigger IRQ.<br>0: No triggered IRQ signal.<br>1: Trigger IRQ signal.<br>**NOTES**<br>1. An MCS-channel triggers an IRQ by writing value 1 to bit IRQ. Writing a value 0 to this bit does not cancel the IRQ, and thus has no effect.<br>2. This bit mirrors bit 0 of the register GTM0MCSixIRQNOTIFY.<br>3. The IRQ bit can only be cleared by CPU, by writing a 1 to the corresponding GTM0MCSixIRQNOTIFY register (see **Section 25.13.9.6, GTM0MCSixIRQNOTIFY (i = 0, x = 0 to 8, i = 1, x = 0 to 5)** ).<br>4. An MCS-channel can read the IRQ bit in order to determine the current state of the IRQ handling. The MCS-channel reads a value 1 if an IRQ was released but not cleared by CPU. If an MCS-channel reads a value 0 no IRQ was released or it has been cleared by CPU. |
| 0 | EN | Enable current MCS-channel.<br>0: Disable current MCS-channel.<br>1: Enable current MCS-channel. |

**NOTE**

Writing to bits of the register STA with instructions that do implicitly a read-modify-write operation (e.g. “ANDL STA, $FFFFFE_H$” or “OR STA, R0”) is dangerous, since writing back the possibly modified content of the read access (which reflects status information) may cause undesirable results. A secure way for writing to bits of the register STA is to use instructions that do not read the content of STA (e.g. “MOVL STA, 0x0 or MOV STA, R1”).

### 25.13.7.3 Register ACB

**Access:** This register can be read in 24-bit units.

**Address:** —

**Value after reset:** $0000\ 0000_H$

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | — | — | — | ACB4 | ACB3 | ACB2 | ACB1 | ACB0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 25.144 ACB Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 23 to 5 | Reserved | These bits are always read as 0. |
| 4 | ACB4 | ARU Control bit 4.<br>**NOTE**<br>This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 52 of the ARU word. |
| 3 | ACB3 | ARU Control bit 3.<br>**NOTE**<br>This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 51 of the ARU word. |
| 2 | ACB2 | ARU Control bit 2.<br>**NOTE**<br>This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 50 of the ARU word. |
| 1 | ACB1 | ARU Control bit 1.<br>**NOTE**<br>This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 49 of the ARU word. |
| 0 | ACB0 | ARU Control bit 0.<br>**NOTE**<br>This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 48 of the ARU word. |

### 25.13.7.4 Register CTRG

**Access:** This register can be read in 24-bit units.

**Address:** —

**Value after reset:** 0000 $0000_H$

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | TRG15 | TRG14 | TRG13 | TRG12 | TRG11 | TRG10 | TRG9 | TRG8 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TRG7 | TRG6 | TRG5 | TRG4 | TRG3 | TRG2 | TRG1 | TRG0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 25.145 CTRG Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 23 to 16 | Reserved | These bits are always read as 0. |
| 15 | TRG15 | Trigger bit 15<br>READ access:<br>State of current trigger bit TRG15 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH7_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG15 |
| 14 | TRG14 | Trigger bit 14<br>READ access:<br>State of current trigger bit TRG14 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH6_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG14 |
| 13 | TRG13 | Trigger bit 13<br>READ access:<br>State of current trigger bit TRG13 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH5_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG13 |
| 12 | TRG12 | Trigger bit 12<br>READ access:<br>State of current trigger bit TRG12 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH4_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG12 |

**Table 25.145 CTRG Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | TRG11 | Trigger bit 11<br>READ access:<br>State of current trigger bit TRG11 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH3_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG11 |
| 10 | TRG10 | Trigger bit 10<br>READ access:<br>State of current trigger bit TRG10 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH2_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG10 |
| 9 | TRG9 | Trigger bit 9<br>READ access:<br>State of current trigger bit TRG9 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH1_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG9 |
| 8 | TRG8 | Trigger bit 8<br>READ access:<br>State of current trigger bit TRG8 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH0_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG8 |
| 7 | TRG7 | Trigger bit 7<br>READ access:<br>State of current trigger bit TRG7 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH7_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG7 |
| 6 | TRG6 | Trigger bit 6<br>READ access:<br>State of current trigger bit TRG6 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH6_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG6 |
| 5 | TRG5 | Trigger bit 5<br>READ access:<br>State of current trigger bit TRG5 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH5_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG5 |
| 4 | TRG4 | Trigger bit 4<br>READ access:<br>State of current trigger bit TRG4 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH4_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG4 |

**Table 25.145 CTRG Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 | TRG3 | Trigger bit 3<br>READ access:<br>State of current trigger bit TRG3 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH3_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG3 |
| 2 | TRG2 | Trigger bit 2<br>READ access:<br>State of current trigger bit TRG2 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH2_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG2 |
| 1 | TRG1 | Trigger bit 1<br>READ access:<br>State of current trigger bit TRG1 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH1_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG1 |
| 0 | TRG0 | Trigger bit 0<br>READ access:<br>State of current trigger bit TRG0 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH0_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG0 |

**NOTES**

1. The result of a read access to this register differs in dependency of the bit field EN_TIM_FOUT of register GTM0MCSiCTRLSTAT.
2. The trigger bits TRGx (x = 0 to 15) are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the STRG register, in the case of an MCS-channel or the GTM0MCSiSTRG register in the case of the CPU. Clearing a trigger bit can be performed with the CTRG register, in the case of an MCS-channel or the GTM0MCSiCTRG register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS-channels or the CPU. An MCS-channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.
3. Besides setting the trigger bits with register STRG/GTM0MCSiSTRG, the k-th trigger bit TRGk can also be set by the external capture event that is enabled by the k-th bit of register GTM0GTMEXTCAPENi. If bit k bit is disabled, the k-th trigger bit TRGk can only be set by MCS or CPU.

### 25.13.7.5 Register STRG

**Access:** This register can be read in 24-bit units.

**Address:** —

**Value after reset:** $00\ 0000_H$

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | TRG15 | TRG14 | TRG13 | TRG12 | TRG11 | TRG10 | TRG9 | TRG8 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TRG7 | TRG6 | TRG5 | TRG4 | TRG3 | TRG2 | TRG1 | TRG0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 25.146 STRG Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 23 to 16 | Reserved | These bits are always read as 0. |
| 15 | TRG15 | Trigger bit 15<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 14 | TRG14 | Trigger bit 14<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 13 | TRG13 | Trigger bit 13<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 12 | TRG12 | Trigger bit 12<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 11 | TRG11 | Trigger bit 11<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 10 | TRG10 | Trigger bit 10<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 9 | TRG9 | Trigger bit 9<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 8 | TRG8 | Trigger bit 8<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 7 | TRG7 | Trigger bit 7<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |

**Table 25.146 STRG Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 | TRG6 | Trigger bit 6<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 5 | TRG5 | Trigger bit 5<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 4 | TRG4 | Trigger bit 4<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 3 | TRG3 | Trigger bit 3<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 2 | TRG2 | Trigger bit 2<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 1 | TRG1 | Trigger bit 1<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 0 | TRG0 | Trigger bit 0<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |

**NOTES**

1. The trigger bits TRGx (x = 0 to 15) are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the STRG register, in the case of an MCS-channel or the GTM0MCSiSTRG register in the case of the CPU. Clearing a trigger bit can be performed with the CTRG register, in the case of an MCS-channel or the GTM0MCSiCTRG register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS-channels or the CPU. An MCS-channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.
2. Besides setting the trigger bits with register STRG/GTM0MCSiSTRG, the k-th trigger bit TRGk can also be set by the external capture event that is enabled by the k-th bit of register GTM0GTMEXTCAPENi. If bit k bit is disabled, the k-th trigger bit TRGk can only be set by MCS or CPU.

### 25.13.7.6 Register TBU_TS0

**Access:** This register can be read in 24-bit units.

**Address:** —

**Value after reset:** $00\ 0000_H$

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | TS | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | TS | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TS | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 25.147 TBU_TS0 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 23 to 0 | TS | Current TBU time stamp 0. |

**NOTE**

Any write access to a time base register discards the written data. A write access to a time base register may be used to destroy an unused 24-bit data word of an ARU read transfer.

### 25.13.7.7 Register TBU_TS1

**Access:** This register can be read in 24-bit units.

**Address:** —

**Value after reset:** $00\ 0000_H$

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | TS | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | TS | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TS | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 25.148 TBU_TS1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 23 to 0 | TS | Current TBU time stamp 1. |

**NOTE**

Any write access to a time base register discards the written data. A write access to a time base register may be used to destroy an unused 24-bit data word of an ARU read transfer.

### 25.13.7.8 Register MHB

**Access:** This register can be read/written in 24-bit units.

**Address:** —

**Value after reset:** $00\ 0000_H$

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DATA | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.149 MHB Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 23 to 8 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 7 to 0 | DATA | High Byte of a memory transfer. |

## 25.13.8 MCS Configuration Registers Overview

The MCS Configuration registers of the MCS module are accessible by the AEI bus interface. Some of these registers simply mirror MCS Internal registers to the AEI. Details can be found in the table below and in the individual register descriptions.

**Table 25.150 Register list**

| Symbol | Register Name | Details in Section |
|---|---|---|
| GTM0MCSixCTRL | MCS Channel control register (x = 0 to 8)<br>Most bits mirror the internal task specific register STA. | **25.13.9.1** |
| GTM0MCSixACB | MCS Channel ACB register (x = 0 to 8)<br>The Register mirrors the internal task specific register ACB. | **25.13.9.4** |
| GTM0MCSixMHB | Memory High Byte register (x = 0 to 8)<br>The Register mirrors the internal task specific register MHB. | **25.13.9.5** |
| GTM0MCSixPC | MCS Channel Program counter register (x = 0 to 8) | **25.13.9.2** |
| GTM0MCSixRy | MCS Channel GPRx registers (x = 0 to 8; y = 0 to 7). These registers mirror the internal task specific registers R[y]. | **25.13.9.3** |
| GTM0MCSixIRQNOTIFY | MCS Channel x interrupt notification register (x = 0 to 8) | **25.13.9.6** |
| GTM0MCSixIRQEN | MCS Channel x interrupt enable register (x = 0 to 8) | **25.13.9.7** |
| GTM0MCSixIRQFORCINT | MCS Channel x software interrupt generation register (x = 0 to 8) | **25.13.9.8** |
| GTM0MCSixIRQMODE | IRQ mode configuration register (x = 0 to 8) | **25.13.9.9** |
| GTM0MCSixEIRQEN | MCS Channel x error interrupt enable register (x = 0 to T–1) | **25.13.9.10** |
| GTM0MCSiCTRLSTAT | MCS Control and Status register | **25.13.9.11** |
| GTM0MCSiCTRG | MCS Clear trigger control register. | **25.13.9.12** |
| GTM0MCSiSTRG | MCS Set trigger control register | **25.13.9.13** |
| GTM0MCSiRESET | MCS Channel reset register | **25.13.9.14** |
| GTM0MCSiERR | MCS Error register | **25.13.9.15** |
| GTM0MCSiCAT | Cancel ARU transfer register. | **25.13.9.16** |
| GTM0MCS0CWT | Cancel WURM instruction. | **25.13.9.17** |

## 25.13.9 MCS Configuration Register Description

### 25.13.9.1 GTM0MCSixCTRL (i = 0, x = 0 to 8, i = 1, x = 0 to 5)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0MCS00CTRL: <GTM_base> + $30020_H$.
GTM0MCS01CTRL: <GTM_base> + $300A0_H$.
GTM0MCS02CTRL: <GTM_base> + $30120_H$.
GTM0MCS03CTRL: <GTM_base> + $301A0_H$.
GTM0MCS04CTRL: <GTM_base> + $30220_H$.
GTM0MCS05CTRL: <GTM_base> + $302A0_H$.
GTM0MCS06CTRL: <GTM_base> + $30320_H$.
GTM0MCS07CTRL: <GTM_base> + $303A0_H$.
GTM0MCS08CTRL: <GTM_base> + $30420_H$.
GTM0MCS10CTRL: <GTM_base> + $31020_H$
GTM0MCS11CTRL: <GTM_base> + $310A0_H$
GTM0MCS12CTRL: <GTM_base> + $31120_H$
GTM0MCS13CTRL: <GTM_base> + $311A0_H$
GTM0MCS14CTRL: <GTM_base> + $31220_H$
GTM0MCS15CTRL: <GTM_base> + $312A0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | SP_CNT | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SAT | CWT | CAT | N | V | Z | CY | — | ERR | IRQ | EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 25.151 GTM0MCSixCTRL Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 18 to 16 | SP_CNT | Stack pointer counter value.<br>Actual stack depth of channel. The bit field is incremented on behalf of a CALL or PUSH instruction and decremented on behalf of a RET or POP instruction. The MCS channel STK_ERR_IRQ is raised, when an overflow or underflow is detected on this bit field. |
| 15 to 11 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 10 | SAT | Successful ARU transfer bit.<br>0: Non-blocking ARU transfer failed due to missing data.<br>1: Non-blocking ARU transfer finished successfully.<br>**NOTE**<br>This bit is read only and it mirrors the internal state of the ARU transfer status flag SAT. |
| 9 | CWT | Cancel WURM instruction state.<br>**NOTE**<br>This bit is read only and it mirrors the internal cancel WURM instruction status flag CWT. |
| 8 | CAT | Cancel ARU transfer state.<br>**NOTE**<br>This bit is read only and it mirrors the internal state of the ARU transfer status flag SAT. |

**Table 25.151 GTM0MCSixCTRL Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | N | Negative bit state.<br>**NOTE**<br>This bit is read only and it mirrors the internal cancel ARU transfer status flag CAT. |
| 6 | V | Overflow bit state.<br>**NOTE**<br>This bit is read only and it mirrors the internal carry flag V. |
| 5 | Z | Zero bit state.<br>**NOTE**<br>This bit is read only and it mirrors the internal zero flag Z. |
| 4 | CY | Carry bit state.<br>**NOTE**<br>This bit is read only and it mirrors the internal carry flag CY. |
| 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 | ERR | Error state.<br>0: No error signal pending in MCS-channel x.<br>1: Error signal is pending in MCS-channel x.<br>**NOTE**<br>This bit is read only and it mirrors the internal error state. |
| 1 | IRQ | Interrupt state.<br>0: No interrupt pending in MCS-channel x.<br>1: Interrupt is pending in MCS-channel x.<br>**NOTE**<br>This bit is read only and it mirrors the internal IRQ state. |
| 0 | EN | Enable MCS-channel<br>0: Disable current MCS-channel.<br>1: Enable current MCS-channel.<br>**NOTES**<br>1. Enabling or disabling of an MCS-channel is synchronized to the ending of an instruction and thus it may take several clock cycles, e.g. active memory transfers or pending WURM transfers have to be finished before disabling the MCS-channel. The internal state of a channel can be obtained by reading the bit EN.<br>2. To disable an MCS channel reliably the EN bit should be cleared followed by setting the CAT and CWT bit in order to cancel any pending WURM or ARU instructions.<br>3. The EN bit is write protected during RAM reset phase. |

### 25.13.9.2 GTM0MCSixPC (i = 0, x = 0 to 8, i = 1, x = 0 to 5)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0MCS00PC: <GTM_base> + $30040_H$.
GTM0MCS01PC: <GTM_base> + $300C0_H$.
GTM0MCS02PC: <GTM_base> + $30140_H$.
GTM0MCS03PC: <GTM_base> + $301C0_H$.
GTM0MCS04PC: <GTM_base> + $30240_H$.
GTM0MCS05PC: <GTM_base> + $302C0_H$.
GTM0MCS06PC: <GTM_base> + $30340_H$.
GTM0MCS07PC: <GTM_base> + $303C0_H$.
GTM0MCS08PC: <GTM_base> + $30440_H$.
GTM0MCS10PC: <GTM_base> + $31040_H$
GTM0MCS11PC: <GTM_base> + $310C0_H$
GTM0MCS12PC: <GTM_base> + $31140_H$
GTM0MCS13PC: <GTM_base> + $311C0_H$
GTM0MCS14PC: <GTM_base> + $31240_H$
GTM0MCS15PC: <GTM_base> + $312C0_H$

**Value after reset:** 0000 0000+4*$x_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PC | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.152 GTM0MCSixPC (x = 0 to 8) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 0 | PC | Current Program Counter.<br>**NOTES**<br>1. The program counter is only writable if the corresponding MCS-channel is disabled. The bits 0 and 1 are always written as zeros.<br>2. The actual width of the program counter depends on the MCS configuration. The actual width is RAW+USR+2 bits meaning that only the bits 0 to RAW+USR+1 are available and the other bits (RAW+USR+2 to 31) are reserved. |

### 25.13.9.3 GTM0MCSixRy (x = 0 to 8, y = 0 to 7)

**Access:** This register can be read/written in 32-bit units.

**Address:** See **Table 25.154**

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DATA | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.153 GTM0MCSixRy (x = 0 to 8, y = 0 to 7) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 0 | DATA | Data of general purpose register R[y]. |

**NOTES**

1. This register is the same as described in **Section 25.13.7.1, Register R[y] (y = 0 to 7)**.
2. For the register GTM0MCSixR6 **Section 25.13.7.1, Register R[y] (y = 0 to 7)** an additional write protection during an active ARDI or NARDI instruction is applied.

**Table 25.154 GTM0MCSixRy Register Address list (1/3)**

| Instance name[i] | Channel number[x] | Symbol | Address | Instance name[i] | channel number[x] | Symbol | Address |
|---|---|---|---|---|---|---|---|
| MCS0 | 0 | GTM0MCS00R0 | <GTM0_base> + 30000$_H$ | MCS1 | 0 | GTM0MCS10R0 | <GTM0_base> + 31000$_H$ |
| MCS0 | 0 | GTM0MCS00R1 | <GTM0_base> + 30004$_H$ | MCS1 | 0 | GTM0MCS10R1 | <GTM0_base> + 31004$_H$ |
| MCS0 | 0 | GTM0MCS00R2 | <GTM0_base> + 30008$_H$ | MCS1 | 0 | GTM0MCS10R2 | <GTM0_base> + 31008$_H$ |
| MCS0 | 0 | GTM0MCS00R3 | <GTM0_base> + 3000C$_H$ | MCS1 | 0 | GTM0MCS10R3 | <GTM0_base> + 3100C$_H$ |
| MCS0 | 0 | GTM0MCS00R4 | <GTM0_base> + 30010$_H$ | MCS1 | 0 | GTM0MCS10R4 | <GTM0_base> + 31010$_H$ |
| MCS0 | 0 | GTM0MCS00R5 | <GTM0_base> + 30014$_H$ | MCS1 | 0 | GTM0MCS10R5 | <GTM0_base> + 31014$_H$ |
| MCS0 | 0 | GTM0MCS00R6 | <GTM0_base> + 30018$_H$ | MCS1 | 0 | GTM0MCS10R6 | <GTM0_base> + 31018$_H$ |
| MCS0 | 0 | GTM0MCS00R7 | <GTM0_base> + 3001C$_H$ | MCS1 | 0 | GTM0MCS10R7 | <GTM0_base> + 3101C$_H$ |
| MCS0 | 1 | GTM0MCS01R0 | <GTM0_base> + 30080$_H$ | MCS1 | 1 | GTM0MCS11R0 | <GTM0_base> + 31080$_H$ |
| MCS0 | 1 | GTM0MCS01R1 | <GTM0_base> + 30084$_H$ | MCS1 | 1 | GTM0MCS11R1 | <GTM0_base> + 31084$_H$ |
| MCS0 | 1 | GTM0MCS01R2 | <GTM0_base> + 30088$_H$ | MCS1 | 1 | GTM0MCS11R2 | <GTM0_base> + 31088$_H$ |
| MCS0 | 1 | GTM0MCS01R3 | <GTM0_base> + 3008C$_H$ | MCS1 | 1 | GTM0MCS11R3 | <GTM0_base> + 3108C$_H$ |
| MCS0 | 1 | GTM0MCS01R4 | <GTM0_base> + 30090$_H$ | MCS1 | 1 | GTM0MCS11R4 | <GTM0_base> + 31090$_H$ |
| MCS0 | 1 | GTM0MCS01R5 | <GTM0_base> + 30094$_H$ | MCS1 | 1 | GTM0MCS11R5 | <GTM0_base> + 31094$_H$ |
| MCS0 | 1 | GTM0MCS01R6 | <GTM0_base> + 30098$_H$ | MCS1 | 1 | GTM0MCS11R6 | <GTM0_base> + 31098$_H$ |
| MCS0 | 1 | GTM0MCS01R7 | <GTM0_base> + 3009C$_H$ | MCS1 | 1 | GTM0MCS11R7 | <GTM0_base> + 3109C$_H$ |
| MCS0 | 2 | GTM0MCS02R0 | <GTM0_base> + 30100$_H$ | MCS1 | 2 | GTM0MCS12R0 | <GTM0_base> + 31100$_H$ |
| MCS0 | 2 | GTM0MCS02R1 | <GTM0_base> + 30104$_H$ | MCS1 | 2 | GTM0MCS12R1 | <GTM0_base> + 31104$_H$ |
| MCS0 | 2 | GTM0MCS02R2 | <GTM0_base> + 30108$_H$ | MCS1 | 2 | GTM0MCS12R2 | <GTM0_base> + 31108$_H$ |

Table 25.154 GTM0MCSixRy Register Address list (2/3)

| Instance name[i] | Channel number[x] | Symbol | Address | Instance name[i] | channel number[x] | Symbol | Address |
|---|---|---|---|---|---|---|---|
| MCS0 | 2 | GTM0MCS02R3 | <GTM0_base> + 3010C$_H$ | MCS1 | 2 | GTM0MCS12R3 | <GTM0_base> + 3110C$_H$ |
| MCS0 | 2 | GTM0MCS02R4 | <GTM0_base> + 30110$_H$ | MCS1 | 2 | GTM0MCS12R4 | <GTM0_base> + 31110$_H$ |
| MCS0 | 2 | GTM0MCS02R5 | <GTM0_base> + 30114$_H$ | MCS1 | 2 | GTM0MCS12R5 | <GTM0_base> + 31114$_H$ |
| MCS0 | 2 | GTM0MCS02R6 | <GTM0_base> + 30118$_H$ | MCS1 | 2 | GTM0MCS12R6 | <GTM0_base> + 31118$_H$ |
| MCS0 | 2 | GTM0MCS02R7 | <GTM0_base> + 3011C$_H$ | MCS1 | 2 | GTM0MCS12R7 | <GTM0_base> + 3111C$_H$ |
| MCS0 | 3 | GTM0MCS03R0 | <GTM0_base> + 30180$_H$ | MCS1 | 3 | GTM0MCS13R0 | <GTM0_base> + 31180$_H$ |
| MCS0 | 3 | GTM0MCS03R1 | <GTM0_base> + 30184$_H$ | MCS1 | 3 | GTM0MCS13R1 | <GTM0_base> + 31184$_H$ |
| MCS0 | 3 | GTM0MCS03R2 | <GTM0_base> + 30188$_H$ | MCS1 | 3 | GTM0MCS13R2 | <GTM0_base> + 31188$_H$ |
| MCS0 | 3 | GTM0MCS03R3 | <GTM0_base> + 3018C$_H$ | MCS1 | 3 | GTM0MCS13R3 | <GTM0_base> + 3118C$_H$ |
| MCS0 | 3 | GTM0MCS03R4 | <GTM0_base> + 30190$_H$ | MCS1 | 3 | GTM0MCS13R4 | <GTM0_base> + 31190$_H$ |
| MCS0 | 3 | GTM0MCS03R5 | <GTM0_base> + 30194$_H$ | MCS1 | 3 | GTM0MCS13R5 | <GTM0_base> + 31194$_H$ |
| MCS0 | 3 | GTM0MCS03R6 | <GTM0_base> + 30198$_H$ | MCS1 | 3 | GTM0MCS13R6 | <GTM0_base> + 31198$_H$ |
| MCS0 | 3 | GTM0MCS03R7 | <GTM0_base> + 3019C$_H$ | MCS1 | 3 | GTM0MCS13R7 | <GTM0_base> + 3119C$_H$ |
| MCS0 | 4 | GTM0MCS04R0 | <GTM0_base> + 30200$_H$ | MCS1 | 4 | GTM0MCS14R0 | <GTM0_base> + 31200$_H$ |
| MCS0 | 4 | GTM0MCS04R1 | <GTM0_base> + 30204$_H$ | MCS1 | 4 | GTM0MCS14R1 | <GTM0_base> + 31204$_H$ |
| MCS0 | 4 | GTM0MCS04R2 | <GTM0_base> + 30208$_H$ | MCS1 | 4 | GTM0MCS14R2 | <GTM0_base> + 31208$_H$ |
| MCS0 | 4 | GTM0MCS04R3 | <GTM0_base> + 3020C$_H$ | MCS1 | 4 | GTM0MCS14R3 | <GTM0_base> + 3120C$_H$ |
| MCS0 | 4 | GTM0MCS04R4 | <GTM0_base> + 30210$_H$ | MCS1 | 4 | GTM0MCS14R4 | <GTM0_base> + 31210$_H$ |
| MCS0 | 4 | GTM0MCS04R5 | <GTM0_base> + 30214$_H$ | MCS1 | 4 | GTM0MCS14R5 | <GTM0_base> + 31214$_H$ |
| MCS0 | 4 | GTM0MCS04R6 | <GTM0_base> + 30218$_H$ | MCS1 | 4 | GTM0MCS14R6 | <GTM0_base> + 31218$_H$ |
| MCS0 | 4 | GTM0MCS04R7 | <GTM0_base> + 3021C$_H$ | MCS1 | 4 | GTM0MCS14R7 | <GTM0_base> + 3121C$_H$ |
| MCS0 | 5 | GTM0MCS05R0 | <GTM0_base> + 30280$_H$ | MCS1 | 5 | GTM0MCS15R0 | <GTM0_base> + 31280$_H$ |
| MCS0 | 5 | GTM0MCS05R1 | <GTM0_base> + 30284$_H$ | MCS1 | 5 | GTM0MCS15R1 | <GTM0_base> + 31284$_H$ |
| MCS0 | 5 | GTM0MCS05R2 | <GTM0_base> + 30288$_H$ | MCS1 | 5 | GTM0MCS15R2 | <GTM0_base> + 31288$_H$ |
| MCS0 | 5 | GTM0MCS05R3 | <GTM0_base> + 3028C$_H$ | MCS1 | 5 | GTM0MCS15R3 | <GTM0_base> + 3128C$_H$ |
| MCS0 | 5 | GTM0MCS05R4 | <GTM0_base> + 30290$_H$ | MCS1 | 5 | GTM0MCS15R4 | <GTM0_base> + 31290$_H$ |
| MCS0 | 5 | GTM0MCS05R5 | <GTM0_base> + 30294$_H$ | MCS1 | 5 | GTM0MCS15R5 | <GTM0_base> + 31294$_H$ |
| MCS0 | 5 | GTM0MCS05R6 | <GTM0_base> + 30298$_H$ | MCS1 | 5 | GTM0MCS15R6 | <GTM0_base> + 31298$_H$ |
| MCS0 | 5 | GTM0MCS05R7 | <GTM0_base> + 3029C$_H$ | MCS1 | 5 | GTM0MCS15R7 | <GTM0_base> + 3129C$_H$ |
| MCS0 | 6 | GTM0MCS06R0 | <GTM0_base> + 30300$_H$ | — | — | — | — |
| MCS0 | 6 | GTM0MCS06R1 | <GTM0_base> + 30304$_H$ | — | — | — | — |
| MCS0 | 6 | GTM0MCS06R2 | <GTM0_base> + 30308$_H$ | — | — | — | — |
| MCS0 | 6 | GTM0MCS06R3 | <GTM0_base> + 3030C$_H$ | — | — | — | — |
| MCS0 | 6 | GTM0MCS06R4 | <GTM0_base> + 30310$_H$ | — | — | — | — |
| MCS0 | 6 | GTM0MCS06R5 | <GTM0_base> + 30314$_H$ | — | — | — | — |
| MCS0 | 6 | GTM0MCS06R6 | <GTM0_base> + 30318$_H$ | — | — | — | — |
| MCS0 | 6 | GTM0MCS06R7 | <GTM0_base> + 3031C$_H$ | — | — | — | — |
| MCS0 | 7 | GTM0MCS07R0 | <GTM0_base> + 30380$_H$ | — | — | — | — |
| MCS0 | 7 | GTM0MCS07R1 | <GTM0_base> + 30384$_H$ | — | — | — | — |
| MCS0 | 7 | GTM0MCS07R2 | <GTM0_base> + 30388$_H$ | — | — | — | — |
| MCS0 | 7 | GTM0MCS07R3 | <GTM0_base> + 3038C$_H$ | — | — | — | — |
| MCS0 | 7 | GTM0MCS07R4 | <GTM0_base> + 30390$_H$ | — | — | — | — |
| MCS0 | 7 | GTM0MCS07R5 | <GTM0_base> + 30394$_H$ | — | — | — | — |
| MCS0 | 7 | GTM0MCS07R6 | <GTM0_base> + 30398$_H$ | — | — | — | — |
| MCS0 | 7 | GTM0MCS07R7 | <GTM0_base> + 3039C$_H$ | — | — | — | — |
| MCS0 | 8 | GTM0MCS08R0 | <GTM0_base> + 30400$_H$ | — | — | — | — |
| MCS0 | 8 | GTM0MCS08R1 | <GTM0_base> + 30404$_H$ | — | — | — | — |

**Table 25.154 GTM0MCSixRy Register Address list (3/3)**

| Instance name[i] | Channel number[x] | Symbol | Address | Instance name[i] | channel number[x] | Symbol | Address |
|---|---|---|---|---|---|---|---|
| MCS0 | 8 | GTM0MCS08R2 | <GTM0_base> + $30408_H$ | — | — | — | — |
| MCS0 | 8 | GTM0MCS08R3 | <GTM0_base> + $3040C_H$ | — | — | — | — |
| MCS0 | 8 | GTM0MCS08R4 | <GTM0_base> + $30410_H$ | — | — | — | — |
| MCS0 | 8 | GTM0MCS08R5 | <GTM0_base> + $30414_H$ | — | — | — | — |
| MCS0 | 8 | GTM0MCS08R6 | <GTM0_base> + $30418_H$ | — | — | — | — |
| MCS0 | 8 | GTM0MCS08R7 | <GTM0_base> + $3041C_H$ | — | — | — | — |

### 25.13.9.4 GTM0MCSixACB (i = 0, x = 0 to 8, i = 1, x = 0 to 5)

**Access:** This register can be read in 32-bit units.

**Address:**
GTM0MCS00ACB: <GTM_base> + $30024_H$.
GTM0MCS01ACB: <GTM_base> + $300A4_H$.
GTM0MCS02ACB: <GTM_base> + $30124_H$.
GTM0MCS03ACB: <GTM_base> + $301A4_H$.
GTM0MCS04ACB: <GTM_base> + $30224_H$.
GTM0MCS05ACB: <GTM_base> + $302A4_H$.
GTM0MCS06ACB: <GTM_base> + $30324_H$.
GTM0MCS07ACB: <GTM_base> + $303A4_H$.
GTM0MCS08ACB: <GTM_base> + $30424_H$.
GTM0MCS10ACB: <GTM_base> + $31024_H$
GTM0MCS11ACB: <GTM_base> + $310A4_H$
GTM0MCS12ACB: <GTM_base> + $31124_H$
GTM0MCS13ACB: <GTM_base> + $311A4_H$
GTM0MCS14ACB: <GTM_base> + $31224_H$
GTM0MCS15ACB: <GTM_base> + $312A4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | ACB4 | ACB3 | ACB2 | ACB1 | ACB0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.155 GTM0MCSixACB (x = 0 to 8) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | These bits are always read as 0. |
| 4 | ACB4 | ARU Control bit 4.<br>**NOTE**<br>This bit is read only and it mirrors the internal state. |
| 3 | ACB3 | ARU Control bit 3.<br>**NOTE**<br>This bit is read only and it mirrors the internal state. |
| 2 | ACB2 | ARU Control bit 2.<br>**NOTE**<br>This bit is read only and it mirrors the internal state. |
| 1 | ACB1 | ARU Control bit 1.<br>**NOTE**<br>This bit is read only and it mirrors the internal state. |
| 0 | ACB0 | ARU Control bit 0.<br>**NOTE**<br>This bit is read only and it mirrors the internal state. |

### 25.13.9.5 GTM0MCSixMHB (i = 0, x = 0 to 8, i = 1, x = 0 to 5)

**Access:** This register can be read in 32-bit units.

**Address:**
GTM0MCS00MHB: <GTM_base> + $3003C_H$.
GTM0MCS01MHB: <GTM_base> + $300BC_H$.
GTM0MCS02MHB: <GTM_base> + $3013C_H$.
GTM0MCS03MHB: <GTM_base> + $301BC_H$.
GTM0MCS04MHB: <GTM_base> + $3023C_H$.
GTM0MCS05MHB: <GTM_base> + $302BC_H$.
GTM0MCS06MHB: <GTM_base> + $3033C_H$.
GTM0MCS07MHB: <GTM_base> + $303BC_H$.
GTM0MCS08MHB: <GTM_base> + $3043C_H$.
GTM0MCS10MHB: <GTM_base> + $3103C_H$
GTM0MCS11MHB: <GTM_base> + $310BC_H$
GTM0MCS12MHB: <GTM_base> + $3113C_H$
GTM0MCS13MHB: <GTM_base> + $311BC_H$
GTM0MCS14MHB: <GTM_base> + $3123C_H$
GTM0MCS15MHB: <GTM_base> + $312BC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DATA | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.156 GTM0MCSixMHB (x = 0 to 8) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | These bits are always read as 0. |
| 7 to 0 | DATA | Data of memory high bit register MHB. |

### 25.13.9.6 GTM0MCSixIRQNOTIFY (i = 0, x = 0 to 8, i = 1, x = 0 to 5)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0MCS00IRQNOTIFY: <GTM_base> + $30044_H$.GTM0MCS10IRQNOTIFY: <GTM_base> + $31044_H$
GTM0MCS01IRQNOTIFY: <GTM_base> + $300C4_H$.GTM0MCS11IRQNOTIFY: <GTM_base> + $310C4_H$
GTM0MCS02IRQNOTIFY: <GTM_base> + $30144_H$.GTM0MCS12IRQNOTIFY: <GTM_base> + $31144_H$
GTM0MCS03IRQNOTIFY: <GTM_base> + $301C4_H$.GTM0MCS13IRQNOTIFY: <GTM_base> + $311C4_H$
GTM0MCS04IRQNOTIFY: <GTM_base> + $30244_H$.GTM0MCS14IRQNOTIFY: <GTM_base> + $31244_H$
GTM0MCS05IRQNOTIFY: <GTM_base> + $302C4_H$.GTM0MCS15IRQNOTIFY: <GTM_base> + $312C4_H$
GTM0MCS06IRQNOTIFY: <GTM_base> + $30344_H$.
GTM0MCS07IRQNOTIFY: <GTM_base> + $303C4_H$.
GTM0MCS08IRQNOTIFY: <GTM_base> + $30444_H$.

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ERR_IRQ | STK_ERR_IRQ | MCS_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 25.157 GTM0MCSixIRQNOTIFY (x = 0 to 8) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 | ERR_IRQ | MCS channel x ERR interrupt.<br>0: No IRQ released<br>1: MCS-channel ERR IRQ released.<br>**NOTES**<br>1. If the ERR bit of register STA is triggered the ERR_IRQ will also be set.<br>2. This bit will be cleared on a CPU write access with a value ‘1’. A read access leaves the bit unchanged. |
| 1 | STK_ERR_IRQ | Stack counter overflow/underflow of channel x.<br>0: No IRQ released<br>1: A stack counter overflow or underflow occurred<br>**NOTE**<br>This bit will be cleared on a CPU write access with a value ‘1’. A read access leaves the bit unchanged. |
| 0 | MCS_IRQ | Interrupt request by MCS-channel x.<br>0: No IRQ released<br>1: IRQ released by MCS-channel<br>**NOTES**<br>1. This bit will be cleared on a CPU write access with a value ‘1’. A read access leaves the bit unchanged.<br>2. By writing a '1' to this register, the IRQ flag in the MCS channel status register STA is cleared. |

### 25.13.9.7 GTM0MCSixIRQEN (i = 0, x = 0 to 8, i = 1, x = 0 to 5)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0MCS00IRQEN: <GTM_base> + $30048_H$. GTM0MCS10IRQEN: <GTM_base> + $31048_H$
GTM0MCS01IRQEN: <GTM_base> + $300C8_H$. GTM0MCS11IRQEN: <GTM_base> + $310C8_H$
GTM0MCS02IRQEN: <GTM_base> + $30148_H$. GTM0MCS12IRQEN: <GTM_base> + $31148_H$
GTM0MCS03IRQEN: <GTM_base> + $301C8_H$. GTM0MCS13IRQEN: <GTM_base> + $311C8_H$
GTM0MCS04IRQEN: <GTM_base> + $30248_H$. GTM0MCS14IRQEN: <GTM_base> + $31248_H$
GTM0MCS05IRQEN: <GTM_base> + $302C8_H$. GTM0MCS15IRQEN: <GTM_base> + $312C8_H$
GTM0MCS06IRQEN: <GTM_base> + $30348_H$.
GTM0MCS07IRQEN: <GTM_base> + $303C8_H$.
GTM0MCS08IRQEN: <GTM_base> + $30448_H$.

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ERR_IRQ_EN | STK_ERR_IRQ_ | MCS_IRQ_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 25.158 GTM0MCSixIRQEN (x = 0 to 8) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 | ERR_IRQ_EN | MCS channel x ERR_IRQ interrupt enable<br>0: Disable interrupt<br>1: Enable interrupt |
| 1 | STK_ERR_IRQ_ | MCS channel x STK_ERR_IRQ interrupt enable<br>0: Disable interrupt<br>1: Enable interrupt |
| 0 | MCS_IRQ_EN | MCS channel x MCS_IRQ interrupt enable<br>0: Disable interrupt<br>1: Enable interrupt |

### 25.13.9.8 Register GTM0MCSixIRQFORCINT (i = 0, x = 0 to 8, i = 1, x = 0 to 5)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0MCS00IRQFORCINT: <GTM_base> + $3004C_H$. GTM0MCS10IRQFORCINT: <GTM_base> + $3104C_H$
GTM0MCS01IRQFORCINT: <GTM_base> + $300CC_H$.GTM0MCS11IRQFORCINT: <GTM_base> + $310CC_H$
GTM0MCS02IRQFORCINT: <GTM_base> + $3014C_H$. GTM0MCS12IRQFORCINT: <GTM_base> + $3114C_H$
GTM0MCS03IRQFORCINT: <GTM_base> + $301CC_H$.GTM0MCS13IRQFORCINT: <GTM_base> + $311CC_H$
GTM0MCS04IRQFORCINT: <GTM_base> + $3024C_H$. GTM0MCS14IRQFORCINT: <GTM_base> + $3124C_H$
GTM0MCS05IRQFORCINT: <GTM_base> + $302CC_H$.GTM0MCS15IRQFORCINT: <GTM_base> + $312CC_H$
GTM0MCS06IRQFORCINT: <GTM_base> + $3034C_H$.
GTM0MCS07IRQFORCINT: <GTM_base> + $303CC_H$.
GTM0MCS08IRQFORCINT: <GTM_base> + $3044C_H$.

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TRG_ERR_IRQ | TRG_STK_ERRQ | TRG_MCS_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 25.159 GTM0MCSixIRQFORCINT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 | TRG_ERR_IRQ | Trigger IRQ bit in GTM0MCSixIRQNOTIFY register by software<br>0: No interrupt triggering<br>1: Assert corresponding field in GTM0MCSixIRQNOTIFY register<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of register GTM0GTMCTRL |
| 1 | TRG_STK_ERR_IRQ | Trigger IRQ bit in GTM0MCSixIRQNOTIFY register by software<br>0: No interrupt triggering<br>1: Assert corresponding field in GTM0MCSixIRQNOTIFY register<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of register GTM0GTMCTRL |
| 0 | TRG_MCS_IRQ | Trigger IRQ bit in GTM0MCSixIRQNOTIFY register by software<br>0: No interrupt triggering<br>1: Assert corresponding field in GTM0MCSixIRQNOTIFY register<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of register GTM0GTMCTRL |

### 25.13.9.9 GTM0MCSixIRQMODE (i = 0, x = 0 to 8, i = 1, x = 0 to 5)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0MCS00IRQMODE: <GTM_base> + $30050_H$.
GTM0MCS01IRQMODE: <GTM_base> + $300D0_H$.
GTM0MCS02IRQMODE: <GTM_base> + $30150_H$.
GTM0MCS03IRQMODE: <GTM_base> + $301D0_H$.
GTM0MCS04IRQMODE: <GTM_base> + $30250_H$.
GTM0MCS05IRQMODE: <GTM_base> + $302D0_H$.
GTM0MCS06IRQMODE: <GTM_base> + $30350_H$.
GTM0MCS07IRQMODE: <GTM_base> + $303D0_H$.
GTM0MCS08IRQMODE: <GTM_base> + $30450_H$.
GTM0MCS10IRQMODE: <GTM_base> + $31050_H$
GTM0MCS11IRQMODE: <GTM_base> + $310D0_H$
GTM0MCS12IRQMODE: <GTM_base> + $31150_H$
GTM0MCS13IRQMODE: <GTM_base> + $311D0_H$
GTM0MCS14IRQMODE: <GTM_base> + $31250_H$
GTM0MCS15IRQMODE: <GTM_base> + $312D0_H$

**Value after reset:** $0000\ 000X_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IRQ_MODE | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 25.160 GTM0MCSixIRQMODE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1, 0 | IRQ_MODE | IRQ mode selection<br>00: Level mode<br>01: Pulse mode<br>10: Pulse-Notify mode<br>11: Single-Pulse mode<br>**NOTE**<br>The interrupt modes are described in **Section 25.6.5, GTM-IP Interrupt Concept** . |

### 25.13.9.10GTM0MCSixEIRQEN (i = 0, x = 0 to 8, i = 1, x = 0 to 5)

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0MCS00EIRQEN: <GTM_base> + $30054_H$.
GTM0MCS01EIRQEN: <GTM_base> + $300D4_H$.
GTM0MCS02EIRQEN: <GTM_base> + $30154_H$.
GTM0MCS03EIRQEN: <GTM_base> + $301D4_H$.
GTM0MCS04EIRQEN: <GTM_base> + $30254_H$.
GTM0MCS05EIRQEN: <GTM_base> + $302D4_H$.
GTM0MCS06EIRQEN: <GTM_base> + $30354_H$.
GTM0MCS07EIRQEN: <GTM_base> + $303D4_H$.
GTM0MCS08EIRQEN: <GTM_base> + $30454_H$.
GTM0MCS10EIRQEN: <GTM_base> + $31054_H$
GTM0MCS11EIRQEN: <GTM_base> + $310D4_H$
GTM0MCS12EIRQEN: <GTM_base> + $31154_H$
GTM0MCS13EIRQEN: <GTM_base> + $311D4_H$
GTM0MCS14EIRQEN: <GTM_base> + $31254_H$
GTM0MCS15EIRQEN: <GTM_base> + $312D4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | ERR_EIRQ_EN | STK_ERR_EIRQ_EN | MCS_EIRQ_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 25.161 GTM0MCSixEIRQEN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 | ERR_EIRQ_EN | MCS channel x ERR_EIRQ error interrupt enable<br>0: Disable error interrupt<br>1: Enable error interrupt |
| 1 | STK_ERR_EIRQ_EN | MCS channel x STK_ERR_IRQ error interrupt enable<br>0: Disable error interrupt<br>1: Enable error interrupt |
| 1, 0 | MCS_EIRQ_EN | MCS channel x MCS_EIRQ error interrupt enable<br>0: Disable error interrupt<br>1: Enable error interrupt |

### 25.13.9.11 GTM0MCSiCTRLSTAT (i = 0, 1)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0MCS0CTRLSTAT: <GTM_base> + $30064_H$
GTM0MCS1CTRLSTAT: <GTM_base> + $31064_H$

**Value after reset:** $000X\ 000X_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | EN_TIM_FOUT | — | — | ERR_SRC_ID | | — | — | HLT_SP_OFL | RAM_RST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — |
| R/W | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SCD_CH | | | | — | — | — | — | — | — | SCD_MODE | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

**Table 25.162 GTM0MCSiCTRLSTAT Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 25 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 24 | EN_TIM_FOUT | Enable routing of TIM[i]_CH[x]_F_OUT signal.<br>0: Read access to register CTRG/GTM0MCSiCTRG provides state of the internal trigger registers.<br>1: Read access to register CTRG/GTM0MCSiCTRG provides state of the external signal TIM[i]_CH[x]_F_OUT. |
| 23, 22 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 21, 20 | ERR_SRC_ID | Error source identifier.<br>00: No HW generated Error occurred.<br>01: Detected ECC error.<br>10: Detected memory overflow.<br>11: Detected invalid opcode.<br>**NOTE**<br>This register is updated once, if an ECC error, a memory overflow, or an invalid opcode was detected by the MCS. The register is set to its initial value 00 after each write access to an existing ERR bit in the register GTM0MCSiERR. If multiple errors occur, ERR_SRC_ID is holding the first type of error which has occurred. |
| 19, 18 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 17 | HLT_SP_OFL | Halt on stack pointer overflow.<br>0: No halt on MCS-channel stack pointer counter over/underflow.<br>1: MCS-channel is disabled if a stack pointer counter over/underflow occurs. |

**Table 25.162 GTM0MCSiCTRLSTAT Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 16 | RAM_RST | RAM reset bit<br>0: READ: no RAM reset is active / WRITE: do nothing.<br>1: READ: MCS currently resets RAM content / WRITE: trigger RAM reset.<br>**NOTES**<br>1. The RAM reset initializes the memory content with zeros. RAM access and enabling of MCS channels is disabled during active RAM reset.<br>2. This bit is only writable if the bit RF_PROT in register GTM0GTMCTRL is cleared and all MCS-channels are disabled.<br>3. The actual reset values of this bit depends on the silicon vendor configuration. The reset value is 1, if the RAM reset is performed together with the sub module reset, otherwise the reset value is 0. If the reset value is 1, the reset value is changed to 0 by hardware, when the RAM reset finished. |
| 15 to 12 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 11 to 8 | SCD_CH | Channel selection for scheduling algorithm.<br>MCS-channel identifier used by several scheduling modes.<br>**NOTE**<br>The actual width of the bit field SCD_CH is calculated as $\lceil \log2(T+1) \rceil$.<br>Unused most significant bits are reserved and read as zero. |
| 7 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1, 0 | SCD_MODE | Select MCS scheduling mode<br>00: Accelerated Scheduling.<br>01: Round Robin Scheduling.<br>10: Single Priority Scheduling.<br>11: Multiple Priority Scheduling. |

## 25.13.9.12GTM0MCSiCTRG (i =0 ,1)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0MCS0CTRG: <GTM_base> + $30028_H$
GTM0MCS1CTRG: <GTM_base> + $31028_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRG15 | TRG14 | TRG13 | TRG12 | TRG11 | TRG10 | TRG9 | TRG8 | TRG7 | TRG6 | TRG5 | TRG4 | TRG3 | TRG2 | TRG1 | TRG0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.163 GTM0MCSiCTRG Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 | TRG15 | Trigger bit 15<br>READ access:<br>State of current trigger bit TRG15 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH7_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG15 |
| 14 | TRG14 | Trigger bit 14<br>READ access:<br>State of current trigger bit TRG14 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH6_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG14 |
| 13 | TRG13 | Trigger bit 13<br>READ access:<br>State of current trigger bit TRG13 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH5_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG13 |
| 12 | TRG12 | Trigger bit 12<br>READ access:<br>State of current trigger bit TRG12 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH4_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG12 |
| 11 | TRG11 | Trigger bit 11<br>READ access:<br>State of current trigger bit TRG11 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH3_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG11 |

**Table 25.163 GTM0MCSiCTRG Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 10 | TRG10 | Trigger bit 10<br>READ access:<br>State of current trigger bit TRG10 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH2_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG10 |
| 9 | TRG9 | Trigger bit 9<br>READ access:<br>State of current trigger bit TRG9 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH1_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG9 |
| 8 | TRG8 | Trigger bit 8<br>READ access:<br>State of current trigger bit TRG8 if EN_TIM_FOUT = 0<br>State of input signal TIM[i+1]_CH0_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG8 |
| 7 | TRG7 | Trigger bit 7<br>READ access:<br>State of current trigger bit TRG7 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH7_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG7 |
| 6 | TRG6 | Trigger bit<br>READ access:<br>State of current trigger bit TRG6 if EN_TIM_FOUT = 0<br>State of inpu6t signal TIM[i]_CH6_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG6 |
| 5 | TRG5 | Trigger bit 5<br>READ access:<br>State of current trigger bit TRG5 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH5_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG5 |
| 4 | TRG4 | Trigger bit 4<br>READ access:<br>State of current trigger bit TRG4 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH4_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG4 |
| 3 | TRG3 | Trigger bit 3<br>READ access:<br>State of current trigger bit TRG3 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH3_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG3 |

**Table 25.163 GTM0MCSiCTRG Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | TRG2 | Trigger bit 2<br>READ access:<br>State of current trigger bit TRG2 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH2_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG2 |
| 1 | TRG1 | Trigger bit 1<br>READ access:<br>State of current trigger bit TRG1 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH1_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG1 |
| 0 | TRG0 | Trigger bit 0<br>READ access:<br>State of current trigger bit TRG0 if EN_TIM_FOUT = 0<br>State of input signal TIM[i]_CH0_F_OUT if EN_TIM_FOUT = 1<br>WRITE access:<br>0: Do nothing<br>1: Clear trigger bit TRG0 |

**NOTES**

1. The result of a read access to this register differs in dependency of the bit field EN_TIM_FOUT of register GTM0MCSiCTRLSTAT.
2. The trigger bits TRGx (x = 0 to 15) are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the STRG register, in the case of an MCS-channel or the GTM0MCSiSTRG register in the case of the CPU. Clearing a trigger bit can be performed with the CTRG register, in the case of an MCS-channel or the GTM0MCSiCTRG register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS-channels or the CPU. An MCS-channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.
3. Besides setting the trigger bits with register STRG/GTM0MCSiSTRG, the k-th trigger bit TRGk can also be set by the external capture event that is enabled by the k-th bit of register GTM0GTMEXTCAPENi. If bit k bit is disabled, the k-th trigger bit TRGk can only be set by MCS or CPU.
4. In the scheduling modes Accelerated Scheduling and Round Robin Scheduling, a write access to GTM0MCSiCTRG may take up to T + 1 clock cycles, since the write access is scheduled to the next CPU time slot determined by the MCS scheduler. In the modes Single Prioritization Scheduling and Multiple Prioritization Scheduling, no upper limit access time for a write access to GTM0MCSiCTRG can be guaranteed. The High Prioritized tasks have to be disabled in order to guarantee fast write access to GTM0MCSiCTRG.

## 25.13.9.13GTM0MCSiSTRG (i = 0, 1)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0MCS0STRG: <GTM_base> + $3002C_H$
GTM0MCS1STRG: <GTM_base> + $3102C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRG15 | TRG14 | TRG13 | TRG12 | TRG11 | TRG10 | TRG9 | TRG8 | TRG7 | TRG6 | TRG5 | TRG4 | TRG3 | TRG2 | TRG1 | TRG0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.164 GTM0MCSiSTRG Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 | TRG15 | Trigger bit 15<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 14 | TRG14 | Trigger bit 14<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 13 | TRG13 | Trigger bit 13<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 12 | TRG12 | Trigger bit 12<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 11 | TRG11 | Trigger bit 11<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 10 | TRG10 | Trigger bit 10<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 9 | TRG9 | Trigger bit 9<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 8 | TRG8 | Trigger bit 8<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 7 | TRG7 | Trigger bit 7<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 6 | TRG6 | Trigger bit 6<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 5 | TRG5 | Trigger bit 5<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |

**Table 25.164 GTM0MCSiSTRG Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 | TRG4 | Trigger bit 4<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 3 | TRG3 | Trigger bit 3<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 2 | TRG2 | Trigger bit 2<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 1 | TRG1 | Trigger bit 1<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |
| 0 | TRG0 | Trigger bit 0<br>0: READ: trigger bit is cleared / WRITE: do nothing<br>1: READ: trigger bit is set / WRITE: set trigger bit |

**NOTES**

1. The trigger bits TRGx (x = 0 to 15) are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the STRG register, in the case of an MCS-channel or the GTM0MCSiSTRG register in the case of the CPU. Clearing a trigger bit can be performed with the CTRG register, in the case of an MCS-channel or the GTM0MCSiCTRG register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS-channels or the CPU. An MCS-channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.
2. Besides setting the trigger bits with register STRG/GTM0MCSiSTRG, the k-th trigger bit TRGk can also be set by the external capture event that is enabled by the k-th bit of register GTM0GTMEXTCAPENi. If bit k bit is disabled, the k-th trigger bit TRGk can only be set by MCS or CPU.
3. In the scheduling modes Accelerated Scheduling and Round Robin Scheduling, a write access to GTM0MCSiSTRG may take up to T + 1 clock cycles, since the write access is scheduled to the next CPU time slot determined by the MCS scheduler. In the modes Single Prioritization Scheduling and Multiple Prioritization Scheduling, no upper limit access time for a write access to GTM0MCSiSTRG can be guaranteed. The High Prioritized tasks have to be disabled in order to guarantee fast write access to GTM0MCSiSTRG.

## 25.13.9.14GTM0MCSiRESET (i = 0, 1)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0MCS0RESET: <GTM_base> + $30068_H$
GTM0MCS1RESET: <GTM_base> + $31068_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | RST8 | RST7 | RST6 | RST5 | RST4 | RST3 | RST2 | RST1 | RST0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.165 GTM0MCSiRESET Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 8 | RST8 | Software reset of channel 8<br>0: No action<br>1: Reset channel |
| 7 | RST7 | Software reset of channel 7<br>0: No action<br>1: Reset channel |
| 6 | RST6 | Software reset of channel 6<br>0: No action<br>1: Reset channel |
| 5 | RST5 | Software reset of channel 5<br>0: No action<br>1: Reset channel |
| 4 | RST4 | Software reset of channel 4<br>0: No action<br>1: Reset channel |
| 3 | RST3 | Software reset of channel 3<br>0: No action<br>1: Reset channel |
| 2 | RST2 | Software reset of channel 2<br>0: No action<br>1: Reset channel |
| 1 | RST1 | Software reset of channel 1<br>0: No action<br>1: Reset channel |
| 0 | RST0 | Software reset of channel 0<br>0: No action<br>1: Reset channel |

**NOTES**

1. Only the first T bits of this register (bit 0 to 8) are functionally implemented. The other bits (bit 9 to 31) are reserved bits.
2. The RSTx (x = 0 to 8) bits is cleared automatically after write access of CPU. All channel related registers of channel x are set to their reset values and channel operation is stopped immediately.
3. Channel related registers of channel x are all registers GTM0MCSix*, all MCS internal registers accessible by the corresponding channel, with exception of the common trigger register (accessed by GTM0MCSiCTRG/GTM0MCSiSTRG) and the commonly used general purpose registers GTM0MCSixR4 and GTM0MCSixR5.

## 25.13.9.15GTM0MCSiCAT (i = 0, 1)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0MCS0CAT: <GTM_base> + $3006C_H$
GTM0MCS1CAT: <GTM_base> + $3106C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CAT8 | CAT7 | CAT6 | CAT5 | CAT4 | CAT3 | CAT2 | CAT1 | CAT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.166 GTM0MCSiCAT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 8 | CAT8 | Cancel ARU transfer for channel 8.<br>0: Do nothing.<br>1: Cancel any pending ARU read or write transfer. |
| 7 | CAT7 | Cancel ARU transfer for channel 7.<br>0: Do nothing.<br>1: Cancel any pending ARU read or write transfer. |
| 6 | CAT6 | Cancel ARU transfer for channel 6.<br>0: Do nothing.<br>1: Cancel any pending ARU read or write transfer. |
| 5 | CAT5 | Cancel ARU transfer for channel 5.<br>0: Do nothing.<br>1: Cancel any pending ARU read or write transfer. |
| 4 | CAT4 | Cancel ARU transfer for channel 4.<br>0: Do nothing.<br>1: Cancel any pending ARU read or write transfer. |
| 3 | CAT3 | Cancel ARU transfer for channel 3.<br>0: Do nothing.<br>1: Cancel any pending ARU read or write transfer. |
| 2 | CAT2 | Cancel ARU transfer for channel 2.<br>0: Do nothing.<br>1: Cancel any pending ARU read or write transfer. |
| 1 | CAT1 | Cancel ARU transfer for channel 1.<br>0: Do nothing.<br>1: Cancel any pending ARU read or write transfer. |
| 0 | CAT0 | Cancel ARU transfer for channel 0.<br>0: Do nothing.<br>1: Cancel any pending ARU read or write transfer. |

**NOTES**

1. Only the first T bits of this register (bit 0 to 8) are functionally implemented. The other bits (bit 9 to 31) are reserved bits.
2. The CATx (x = 0 to 8) bit inside the STA register of the corresponding MCS-channel is set and any pending ARU read or write request is cancelled. The MCS-channel resumes with the instruction after the ARU transfer instruction.
3. The CATx (x = 0 to 8) bit is cleared by the corresponding MCS channel, when the channel reaches an ARU read or write instruction.

## 25.13.9.16GTM0MCS0CWT (i = 0, 1)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0MCS0CWT: <GTM_base> + $30070_H$
GTM0MCS1CWT: <GTM_base> + $31070_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CWT8 | CWT7 | CWT6 | CWT5 | CWT4 | CWT3 | CWT2 | CWT1 | CWT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.167 GTM0MCS0CWT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 8 | CWT8 | Cancel WURM instruction for channel 8.<br>0: Do nothing.<br>1: Cancel any pending WURM instruction. |
| 7 | CWT7 | Cancel WURM instruction for channel 7.<br>0: Do nothing.<br>1: Cancel any pending WURM instruction. |
| 6 | CWT6 | Cancel WURM instruction for channel 6.<br>0: Do nothing.<br>1: Cancel any pending WURM instruction. |
| 5 | CWT5 | Cancel WURM instruction for channel 5.<br>0: Do nothing.<br>1: Cancel any pending WURM instruction. |
| 4 | CWT4 | Cancel WURM instruction for channel 4.<br>0: Do nothing.<br>1: Cancel any pending WURM instruction. |
| 3 | CWT3 | Cancel WURM instruction for channel 3.<br>0: Do nothing.<br>1: Cancel any pending WURM instruction. |
| 2 | CWT2 | Cancel WURM instruction for channel 2.<br>0: Do nothing.<br>1: Cancel any pending WURM instruction. |
| 1 | CWT1 | Cancel WURM instruction for channel 1.<br>0: Do nothing.<br>1: Cancel any pending WURM instruction. |
| 0 | CWT0 | Cancel WURM instruction for channel 0.<br>0: Do nothing.<br>1: Cancel any pending WURM instruction. |

**NOTES**

1. Only the first T bits of this register (bit 0 to 8) are functionally implemented. The other bits (bit 9 to 31) are reserved bits.
2. The CWTx (x = 0 to 8) bit inside the STA register of the corresponding MCS-channel is set and any pending WURM instruction is cancelled. The MCS-channel resumes with the instruction after the WURM instruction.
3. The CWTx (x = 0 to 8) bit is cleared by the corresponding MCS channel, when the channel reaches a WURM instruction.

## 25.13.9.17GTM0MCSiERR (i = 0, 1)

**Access:** This register can be read in 32-bit units.

**Address:** GTM0MCS0ERR: <GTM_base> + $3007C_H$
GTM0MCS1ERR: <GTM_base> + $3107C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ERR8 | ERR7 | ERR6 | ERR5 | ERR4 | ERR3 | ERR2 | ERR1 | ERR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.168 GTM0MCSiERR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | These bits are always read as 0. |
| 8 | ERR8 | Error State of MCS-channel 8.<br>0: No error signal.<br>1: Error signal is pending. |
| 7 | ERR7 | Error State of MCS-channel 7.<br>0: No error signal.<br>1: Error signal is pending. |
| 6 | ERR6 | Error State of MCS-channel 6.<br>0: No error signal.<br>1: Error signal is pending. |
| 5 | ERR5 | Error State of MCS-channel 5.<br>0: No error signal.<br>1: Error signal is pending. |
| 4 | ERR4 | Error State of MCS-channel 4.<br>0: No error signal.<br>1: Error signal is pending. |
| 3 | ERR3 | Error State of MCS-channel3 .<br>0: No error signal.<br>1: Error signal is pending. |
| 2 | ERR2 | Error State of MCS-channel 2.<br>0: No error signal.<br>1: Error signal is pending. |
| 1 | ERR1 | Error State of MCS-channel 1.<br>0: No error signal.<br>1: Error signal is pending. |
| 0 | ERR0 | Error State of MCS-channel 0.<br>0: No error signal.<br>1: Error signal is pending. |

**NOTES**

1. Only the first T bits of this register (bit 0 to 8) are functionally implemented. The other bits (bit 9 to 31) are reserved bits.
2. The CPU can read the ERRx (x = 0 to 8) bits in order to determine the current error state of the corresponding MCS-channel x.
3. The error state is also evaluated by the sub module MON, if this module is available.
4. Writing a value 1 to this bit resets the corresponding error state and resets the channel internal ERR bit in the STA and channel CTRL registers. Moreover, each write access to this bit also sets the ERR_SRC_ID bit field of register GTM0MCSiCTRLSTAT to its reset value.

# 25.14 Memory Configuration (MCFG)

## 25.14.1 Overview

The Memory Configuration submodule (MCFG) is an infrastructure module that organizes physical memory blocks and maps them to the RAM ports 0 and 1 of available Multi Channel Sequencer (MCS) modules.

The following parameters are design variables for the MCFG hardware structure that can vary in its range for different devices:

- MAW - Memory address width of a large physical memory block.
- ERM - Enable RAM1 MSB ( 0 - RAM1 MSB disabled,
  1 - RAM1 MSB enabled,

The actual values for these parameters can be obtained from the device specific **Section 25.18, GTM Device 207** and **Section 25.19, GTM Device 208**.

It should be noted that the actual value of the parameter ERM can be obtained by the bit ERM of the register GTM0GTMHWCONF.

Depending on the value of parameter ERM, the MCFG module assumes externally connected physical RAM modules with different sizes. If ERM = 0, MCFG assumes that each MCS instance provides a large physical memory block with $2^{MAW}$ memory locations each 32 bit wide which leads to a RAM module with $2^{MAW+2}$ (byte wise) memory addresses. Further each MCS instance provides a small physical memory block with $2^{MAW-1}$ memory locations each 32 bit wide leading to a RAM module with $2^{MAW+1}$ (byte wise) memory addresses. If ERM = 1, MCFG assumes that each MCS instance provides two large physical memory block each with $2^{MAW}$ memory locations each 32-bit leading to a RAM module with $2^{MAW+2}$ (byte wise) memory addresses.

In order to support different memory sizes for different MCS instances, the MCFG module provides three layout configurations for reorganization of memory pages mapped to the RAM ports of neighboring MCS modules. **Table 25.169, Memory Layout Configurations (ERM = 0)** shows all layout configurations for the case that ERM = 0 and **Table 25.171, Memory Layout Configurations (ERM = 1)** shows the layout configurations for the case that ERM = 1. Each box in these pictures represents a physical memory block.

The layout configuration DEFAULT is always assigning a memory block of size $2^{MAW} \times 32$ bits to MCS RAM port 0. Depending on ERM, RAM port 1 of each MCS is whether assigned to a memory block of size $2^{MAW-1} \times 32$ bits (ERM = 0) or a memory block of size $2^{MAW} \times 32$ bits (ERM = 1).

The layout configuration SWAP is swapping the memory block assigned to RAM port 1 of the current MCS instance with the memory block assigned to RAM port 0 of the successive MCS instance. If ERM = 0, this means that the memory of the current MCS instance is increased by $2^{MAW-1} \times 32$ bits but the memory of the successor is decreased by $2^{MAW-1} \times 32$ bits compared to the DEFAULT configuration. If ERM = 1, the SWAP configuration has no effect on the memory sizes of the individual MCS instances.

The layout configuration BORROW is borrowing the memory block assigned to RAM port 0 of the successive MCS instance for the current instance. This means, the memory of the current MCS module is increased by $2^{MAW} \times 32$ bits but the memory of the successor is decreased by $2^{MAW} \times 32$ bits compared to the DEFAULT configuration.

Considering the order the mentioned MCS modules, it should be noted that the successor of the last MCS instance is the first MCS instance MCS0.

The actual sizes of the memory pages mapped to the MCS RAM ports 0 and 1 depends on the layout configuration for of current instance MCS[i] and the layout configuration of the preceding memory instance MCS[i-1]. The sizes of these memory pages can be obtained by the layout parameters MP0 and MP1, as described in the specification of the MCS.

**Table 25.170, Memory Layout Parameters (ERM = 0)** and **Table 25.172, Memory Layout Parameters (ERM = 1)** summarize the layout parameters MP0 and MP1 of MCS instance MCS[i] for the case that ERM = 0 and ERM = 1. Note that the predecessor of instance MCS0 is last available MCS instance.

The addressing of memory port 0 ranges from 0 to MP0–4 and the addressing of memory page 1 ranges from MP0 to MP1–4.

This document assumes that the GTM implementation embeds 7 MCS instances. However, the actual number of implemented MCS instances can be obtained from [1].

### 25.14.1.1 Memory Layout Configurations (ERM = 0)

**Table 25.169 Memory Layout Configurations (ERM = 0)**

| | DEFAULT | SWAP | BORROW |
|---|---|---|---|
| Configuration for instance MCS[i] | $2^{MAW}$ × 32 bit<br>$2^{MAW-1}$ × 32 bit | $2^{MAW}$ × 32 bit<br>$2^{MAW}$ × 32 bit | $2^{MAW}$ × 32 bit<br>$2^{MAW}$ × 32 bit<br>$2^{MAW-1}$ × 32 bit |
| Configuration for instance MCS[i+1] | $2^{MAW}$ × 32 bit<br>$2^{MAW-1}$ × 32 bit | $2^{MAW-1}$ × 32 bit<br>$2^{MAW-1}$ × 32 bit | $2^{MAW-1}$ × 32 bit |

### 25.14.1.2 Memory Layout Parameters (ERM = 0)

**Table 25.170 Memory Layout Parameters (ERM = 0)**

| | | | Memory Layout Option of preceding MCS instance MCS[i-1] | | |
|---|---|---|---|---|---|
| | | | **DEFAULT** | **SWAP** | **BORROW** |
| Memory Layout Option of current MCS instance MCS[i] | DEFAULT | MP0 | $2^{MAW+2}$ | $2^{MAW+1}$ | 0 |
| | | MP1 | $2^{MAW+2}+2^{MAW+1}$ | $2^{MAW+2}$ | $2^{MAW+1}$ |
| | SWAP | MP0 | $2^{MAW+2}$ | $2^{MAW+1}$ | 0 |
| | | MP1 | $2^{MAW+3}$ | $2^{MAW+2}+2^{MAW+1}$ | $2^{MAW+2}$ |
| | BORROW | MP0 | $2^{MAW+2}$ | $2^{MAW+1}$ | 0 |
| | | MP1 | $2^{MAW+3}+2^{MAW+1}$ | $2^{MAW+3}$ | $2^{MAW+2}+2^{MAW+1}$ |

### 25.14.1.3 Memory Layout Configurations (ERM = 1)

**Table 25.171 Memory Layout Configurations (ERM = 1)**

| | DEFAULT | SWAP | BORROW |
|---|---|---|---|
| Configuration for instance MCS[i] | $2^{MAW}$ × 32 bit<br>$2^{MAW}$ × 32 bit | $2^{MAW}$ × 32 bit<br>$2^{MAW}$ × 32 bit | $2^{MAW}$ × 32 bit<br>$2^{MAW}$ × 32 bit<br>$2^{MAW}$ × 32 bit |
| Configuration for instance MCS[i+1] | $2^{MAW}$ × 32 bit<br>$2^{MAW}$ × 32 bit | $2^{MAW}$ × 32 bit<br>$2^{MAW}$ × 32 bit | $2^{MAW}$ × 32 bit |

#### 25.14.1.4 Memory Layout Parameters (ERM = 1)

Table 25.172 Memory Layout Parameters (ERM = 1)

| | | | Memory Layout Option of preceding MCS instance MCS[i-1] | | |
|---|---|---|---|---|---|
| | | | DEFAULT | SWAP | BORROW |
| Memory Layout Option of current MCS instance MCS[i] | DEFAULT | MP0 | $2^{MAW+2}$ | $2^{MAW+2}$ | 0 |
| | | MP1 | $2^{MAW+3}$ | $2^{MAW+3}$ | $2^{MAW+2}$ |
| | SWAP | MP0 | $2^{MAW+2}$ | $2^{MAW+2}$ | 0 |
| | | MP1 | $2^{MAW+3}$ | $2^{MAW+3}$ | $2^{MAW+2}$ |
| | BORROW | MP0 | $2^{MAW+2}$ | $2^{MAW+2}$ | 0 |
| | | MP1 | $2^{MAW+2}+2^{MAW+3}$ | $2^{MAW+2}+2^{MAW+3}$ | $2^{MAW+3}$ |

### 25.14.2 MCFG Configuration Registers Overview

This section describes the configuration registers of the MCFG submodule.

Table 25.173 Register list

| Register Name | Register Name | Details in Section |
|---|---|---|
| GTM0MCFGCTRL | Memory layout configuration. | **25.14.3.1** |

## 25.14.3 MCFG Configuration Registers

### 25.14.3.1 GTM0MCFGCTRL

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00F40_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | MEM6 | | MEM5 | | MEM4 | | MEM3 | | MEM2 | | MEM1 | | MEM0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.174 GTM0MCFGCTRL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 13, 12 | MEM6 | Configure Memory pages for MCS-instance MCS6.<br>00: DEFAULT configuration<br>01: SWAP configuration<br>10: BORROW configuration<br>11: Reserved |
| 11, 10 | MEM5 | Configure Memory pages for MCS-instance MCS5.<br>00: DEFAULT configuration<br>01: SWAP configuration<br>10: BORROW configuration<br>11: Reserved |
| 9, 8 | MEM4 | Configure Memory pages for MCS-instance MCS4.<br>00: DEFAULT configuration<br>01: SWAP configuration<br>10: BORROW configuration<br>11: Reserved |
| 7, 6 | MEM3 | Configure Memory pages for MCS-instance MCS3.<br>00: DEFAULT configuration<br>01: SWAP configuration<br>10: BORROW configuration<br>11: Reserved |
| 5, 4 | MEM2 | Configure Memory pages for MCS-instance MCS2.<br>00: DEFAULT configuration<br>01: SWAP configuration<br>10: BORROW configuration<br>11: Reserved |
| 3, 2 | MEM1 | Configure Memory pages for MCS-instance MCS1.<br>00: DEFAULT configuration<br>01: SWAP configuration<br>10: BORROW configuration<br>11: Reserved |
| 1, 0 | MEM0 | Configure Memory pages for MCS-instance MCS0.<br>00: DEFAULT configuration<br>01: SWAP configuration<br>10: BORROW configuration<br>11: Reserved |

**NOTE**

It should be noted that the actual GTM-IP implementation may embed less than 7 MCS instances (see **Table 25.2 Sub-Units and Channels**). In this case this register only implements the register bits for available MCS instances.

# 25.15 Interrupt Concentrator Module (ICM)

## 25.15.1 Overview

The Interrupt Concentrator Module (ICM) is used to bundle the GTM-IP interrupt lines of the individual submodules in a reasonable manner into interrupt groups. By this bundling a smaller amount of interrupt lines is visible at the outside of the GTM-IP.

The individual interrupts of the GTM-IP submodules and channels have to be enabled or disabled inside the submodules and channels.

The feed through architecture of bundled interrupt lines is used for the submodules AEI, ARU, CMP, TIM, ATOM and MCS.

To determine the detailed interrupt source the microcontroller has to read the submodule/channel interrupt notification register NOTIFY and serve the channel individual interrupt.

Please note, that the interrupts are only visible inside the ICM and in consequence outside of the GTM-IP, when the interrupt is enabled inside the submodules themselves.

## 25.15.2 Bundling

The GTM-IP submodule individual interrupt sources are connected to the ICM. There, the individual interrupt lines are either feed through and signalled to the outside world or bundled a second time into groups and are then signalled to the outside world.

The ICM interrupt bundling is described in the following sections.

### 25.15.2.1 GTM Infrastructure Interrupt Bundling

The first interrupt group contains interrupts of the infrastructure and safety components of the GTM. This interrupt group includes therefore interrupt lines coming from the AEI, ARU and CMP submodules. In this interrupt group each individual channel of the submodules has its own interrupt line to the outside world.

Thus, the active interrupt line can be used by the CPU to determine the GTM-IP submodule channel that raised the interrupt. The interrupts are also represented in the GTM0ICMIRQG0 register. This register is typically not read by the CPU, but it is readable.

### 25.15.2.2 TIM Interrupt Bundling

Inside this group submodules which handle GTM-IP input signals are treated. This is the case for the TIM[i] submodules. Each TIM submodule channel is able to generate six (6) individual interrupts if enabled inside the TIM channel. This six interrupts are bundled into one interrupt per TIM channel connected to the ICM.

The ICM does no further bundling. Thus, for the GTM-IP 32 interrupt lines TIM[i]_IRQ[y] are provided for the external microcontroller. The channel responsible for the interrupt can be determined by the raised interrupt line.

In addition, the GTM0ICMIRQG2 register is mirror for the TIM submodule channel interrupts and typically not read out by the CPU, but it is readable.

#### 25.15.2.3 MCS Interrupt Bundling

For complex signal output generation, the MCS submodules are used inside the GTM-IP. Each of these MCS submodules could have 32 channels with one interrupt line. This interrupt line is connected to the ICM submodule and is feed through directly to the outside world.

In addition the interrupt line status for the first 8 channels of each MCS are shown in the GTM0ICMIRQG4 and GTM0ICMIRQG5 register.The interrupt line status for all 32 channels of each MCS are shown in the GTM0ICMIRQGMCSiCI register. Typically, the interrupt source is determined by the corresponding interrupt line and the GTM0ICMIRQ4 and GTM0ICMIRQGMCSiCI register are typically not read out by the CPU, but they are readable.

#### 25.15.2.4 ATOM Interrupt Bundling

The interrupts coming from the ATOM[i] submodules are registered in the GTM0ICMIRQG9. Up to four ATOM's are bundled in one ICM register. To identify the ATOM submodule channel where the interrupt occurred, the CPU has to read out the GTM0ICMIRQG9 register first before it goes to the ATOM submodule channel itself.

The GTM0ICMIRQG9 register bits are cleared automatically, when their corresponding interrupt in the submodule channels is cleared.

#### 25.15.2.5 Module Error Interrupt Bundling

The Module Error Interrupt group handles the error interrupts coming from the TIM, CMP submodule of the GTM-IP. The Module Error interrupts are additionally identified in the GTM0ICMIRQGMEIerror interrupt group register. This register is typically not read out by the CPU, but it is readable.

The GTM0ICMIRQGMEI register bits are cleared automatically, when their corresponding error interrupt in the submodule is cleared.

#### 25.15.2.6 TIM Channel Error Interrupt Bundling

The TIM Channel Error Interrupt group handles the error interrupts coming from the TIM channel of the GTM-IP. The TIM Channel Error interrupts are additionally identified for the submodules TIM0 and TIM1 in the GTM0ICMIRQGCEI1 error interrupt group register. This register is typically not read out by the CPU, but it is readable.

The GTM0ICMIRQGCEI1 register bits are cleared automatically, when their corresponding error interrupt in the submodule channel is cleared.

#### 25.15.2.7 MCS Channel Error Interrupt Bundling

The MCS Channel Error Interrupt group handles the error interrupts coming from the MCS channel of the GTM-IP. All 32 MCS Channel Error interrupts are additionally identified for each submodules MCS[i] in the GTM0ICMIRQGMCSiCEI error interrupt group register. The first 8 MCS Channel Error interrupts are additionally identified for the submodules MCS0, MCS1, MCS2 and MCS3 in the GTM0ICMIRQGCEI3 error interrupt group register. These register are typically not read out by the CPU, but they are readable.

The GTM0ICMIRQGMCSiCEI, GTM0ICMIRQGCEI3 register bits are cleared automatically, when their corresponding error interrupt in the submodule channel is cleared.

### 25.15.3 ICM Interrupt Signals

Following table shows the GTM-IP interrupt lines that are visible at the outside of the IP.

**Table 25.175 ICM Interrupt Signals**

| Signal | Description |
|---|---|
| GTM_AEI_IRQ | AEI Shared interrupt |
| GTM_ARU_IRQ[2:0] | [0]: ARU_NEW_DATA0 Interrupt<br>[1]: ARU_NEW_DATA1 Interrupt<br>[2]: ARU_ACC_ACK Interrupt |
| GTM_CMP_IRQ | CMP Shared interrupt |
| GTM_TIM[i]_IRQ[x] | TIM Shared interrupts (i: 0 to number of TIM's-1) (x = 0 to 7) |
| GTM_MCS[i]_IRQ[x] | MCS Interrupt for channel x (x = 0 to 31) (i: 0 to number of MCS's-1) |
| GTM_ATOM[i]_IRQ[x] | ATOM Shared interrupts (i: 0 to number of ATOM's-1) (x = 0 to 7) |
| GTM_ERR_IRQ | GTM Error Interrupt |

### 25.15.4 ICM Configuration Registers Overview

ICM contains following configuration registers:

**Table 25.176 Register list**

| Symbol | Register Name | Details in Section |
|---|---|---|
| GTM0ICMIRQG0 | ICM Interrupt group register covering infrastructural and safety components (ARU, AEI, CMP) | **25.15.5.1** |
| GTM0ICMIRQG2 | ICM Interrupt group register covering TIM0, TIM1 | **25.15.5.2** |
| GTM0ICMIRQG4 | ICM Interrupt group register covering MCS0 to MCS1 submodules | **25.15.5.3** |
| GTM0ICMIRQG9 | ICM Interrupt group register covering GTM-IP output submodules ATOM0, ATOM1, ATOM2 | **25.15.5.4** |
| GTM0ICMIRQGMEI | ICM Interrupt group register for module error interrupt information | **25.15.5.5** |
| GTM0ICMIRQGCEI1 | ICM Interrupt group register 1 for channel error interrupt information | **25.15.5.6** |
| GTM0ICMIRQGCEI3 | ICM Interrupt group register 3 for channel error interrupt information | **25.15.5.7** |
| GTM0ICMIRQGMCS0CI | ICM Interrupt group MCS 0 for Channel Interrupt information | **25.15.5.8** |
| GTM0ICMIRQGMCS0CEI | ICM Interrupt group MCS 0 for Channel Error Interrupt information | **25.15.5.9** |
| GTM0ICMIRQGMCS1CI | ICM Interrupt group MCS 1 for Channel Interrupt information | **25.15.5.10** |
| GTM0ICMIRQGMCS1CEI | ICM Interrupt group MCS 1 for Channel Error Interrupt information | **25.15.5.11** |

## 25.15.5 ICM Configuration Registers Description

### 25.15.5.1 GTM0ICMIRQG0 (GTM Infrastructure Interrupt Group)

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00600_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CMP_IRQ | AEI_IRQ | — | ARU_ACC_ACK_IRQ | ARU_NEW_DATA1_IRQ | ARU_NEW_DATA0_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.177 GTM0ICMIRQG0 (GTM Infrastructure Interrupt Group) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. |
| 5 | CMP_IRQ | CMP shared submodule interrupt. See bit 0. |
| 4 | AEI_IRQ | AEI_IRQ interrupt. See bit 0. |
| 3 | Reserved | This bit is always read as 0. |
| 2 | ARU_ACC_ACK_IRQ | ARU_ACC_ACK interrupt. See bit 0. |
| 1 | ARU_NEW_DATA1_IRQ | ARU_NEW_DATA1 interrupt. See bit 0. |
| 0 | ARU_NEW_DATA0_IRQ | ARU_NEW_DATA0 interrupt<br>0: No interrupt occurred.<br>1: Interrupt was raised by the corresponding submodule.<br>**NOTE**<br>This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. |

### 25.15.5.2 GTM0ICMIRQG2 (TIM Interrupt Group 0)

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00608_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIM1_CH7_IRQ | TIM1_CH6_IRQ | TIM1_CH5_IRQ | TIM1_CH4_IRQ | TIM1_CH3_IRQ | TIM1_CH2_IRQ | TIM1_CH1_IRQ | TIM1_CH0_IRQ | TIM0_CH7_IRQ | TIM0_CH6_IRQ | TIM0_CH5_IRQ | TIM0_CH4_IRQ | TIM0_CH3_IRQ | TIM0_CH2_IRQ | TIM0_CH1_IRQ | TIM0_CH0_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.178 GTM0ICMIRQG2 (TIM Interrupt Group 0) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. |
| 15 | TIM1_CH7_IRQ | TIM1 shared interrupt channel 7. See bit 0. |
| 14 | TIM1_CH6_IRQ | TIM1 shared interrupt channel 6. See bit 0. |
| 13 | TIM1_CH5_IRQ | TIM1 shared interrupt channel 5. See bit 0. |
| 12 | TIM1_CH4_IRQ | TIM1 shared interrupt channel 4. See bit 0. |
| 11 | TIM1_CH3_IRQ | TIM1 shared interrupt channel 3. See bit 0. |
| 10 | TIM1_CH2_IRQ | TIM1 shared interrupt channel 2. See bit 0. |
| 9 | TIM1_CH1_IRQ | TIM1 shared interrupt channel 1. See bit 0. |
| 8 | TIM1_CH0_IRQ | TIM1 shared interrupt channel 0. See bit 0. |
| 7 | TIM0_CH7_IRQ | TIM0 shared interrupt channel 7. See bit 0. |
| 6 | TIM0_CH6_IRQ | TIM0 shared interrupt channel 6. See bit 0. |
| 5 | TIM0_CH5_IRQ | TIM0 shared interrupt channel 5. See bit 0. |
| 4 | TIM0_CH4_IRQ | TIM0 shared interrupt channel 4. See bit 0. |
| 3 | TIM0_CH3_IRQ | TIM0 shared interrupt channel 3. See bit 0. |
| 2 | TIM0_CH2_IRQ | TIM0 shared interrupt channel 2. See bit 0. |
| 1 | TIM0_CH1_IRQ | TIM0 shared interrupt channel 1. See bit 0. |
| 0 | TIM0_CH0_IRQ | TIM0 shared interrupt channel 0.<br>0: No interrupt occurred.<br>1: Interrupt was raised by the corresponding submodule.<br>**NOTES**<br>1. This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.<br>2. When set this bit represents one of the six interrupt sources NEWVALx_IRQ, ECNTOFLx_IRQ, CNTOFLx_IRQ, GPRXOFLx_IRQ, GLITCHDETx_IRQ or TOx_IRQ. |

### 25.15.5.3 GTM0ICMIRQG4 (MCS Interrupt Group 0)

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00610_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCS1_CH7_IRQ | MCS1_CH6_IRQ | MCS1_CH5_IRQ | MCS1_CH4_IRQ | MCS1_CH3_IRQ | MCS1_CH2_IRQ | MCS1_CH1_IRQ | MCS1_CH0_IRQ | MCS0_CH7_IRQ | MCS0_CH6_IRQ | MCS0_CH5_IRQ | MCS0_CH4_IRQ | MCS0_CH3_IRQ | MCS0_CH2_IRQ | MCS0_CH1_IRQ | MCS0_CH0_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.179 GTM0ICMIRQG4 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. |
| 15 | MCS1_CH7_IRQ | MCS1 channel 7 interrupt. See bit 0. |
| 14 | MCS1_CH6_IRQ | MCS1 channel 6 interrupt. See bit 0. |
| 13 | MCS1_CH5_IRQ | MCS1 channel 5 interrupt. See bit 0. |
| 12 | MCS1_CH4_IRQ | MCS1 channel 4 interrupt. See bit 0. |
| 11 | MCS1_CH3_IRQ | MCS1 channel 3 interrupt. See bit 0. |
| 10 | MCS1_CH2_IRQ | MCS1 channel 2 interrupt. See bit 0. |
| 9 | MCS1_CH1_IRQ | MCS1 channel 1 interrupt. See bit 0. |
| 8 | MCS1_CH0_IRQ | MCS1 channel 0 interrupt. See bit 0. |
| 7 | MCS0_CH7_IRQ | MCS0 channel 7 interrupt. See bit 0. |
| 6 | MCS0_CH6_IRQ | MCS0 channel 6 interrupt. See bit 0. |
| 5 | MCS0_CH5_IRQ | MCS0 channel 5 interrupt. See bit 0. |
| 4 | MCS0_CH4_IRQ | MCS0 channel 4 interrupt. See bit 0. |
| 3 | MCS0_CH3_IRQ | MCS0 channel 3 interrupt. See bit 0. |
| 2 | MCS0_CH2_IRQ | MCS0 channel 2 interrupt. See bit 0. |
| 1 | MCS0_CH1_IRQ | MCS0 channel 1 interrupt. See bit 0. |

**Table 25.179 GTM0ICMIRQG4 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | MCS0_CH0_IRQ | MCS0 channel 0 interrupt.<br>0: No interrupt occurred.<br>1: Interrupt was raised by the corresponding submodule.<br>**NOTE**<br>This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. |

### 25.15.5.4 GTM0ICMIRQG9 (ATOM Interrupt Group 0)

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00624_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ATOM2_CH7_IRQ | ATOM2_CH6_IRQ | ATOM2_CH5_IRQ | ATOM2_CH4_IRQ | ATOM2_CH3_IRQ | ATOM2_CH2_IRQ | ATOM2_CH1_IRQ | ATOM2_CH0_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ATOM1_CH7_IRQ | ATOM1_CH6_IRQ | ATOM1_CH5_IRQ | ATOM1_CH4_IRQ | ATOM1_CH3_IRQ | ATOM1_CH2_IRQ | ATOM1_CH1_IRQ | ATOM1_CH0_IRQ | ATOM0_CH7_IRQ | ATOM0_CH6_IRQ | ATOM0_CH5_IRQ | ATOM0_CH4_IRQ | ATOM0_CH3_IRQ | ATOM0_CH2_IRQ | ATOM0_CH1_IRQ | ATOM0_CH0_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.180 GTM0ICMIRQG9 (ATOM Interrupt Group 0) Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. |
| 23 | ATOM2_CH7_IRQ | ATOM2 channel 7 shared interrupt. See bit 0. |
| 22 | ATOM2_CH6_IRQ | ATOM2 channel 6 shared interrupt. See bit 0. |
| 21 | ATOM2_CH5_IRQ | ATOM2 channel 5 shared interrupt. See bit 0. |
| 20 | ATOM2_CH4_IRQ | ATOM2 channel 4 shared interrupt. See bit 0. |
| 19 | ATOM2_CH3_IRQ | ATOM2 channel 3 shared interrupt. See bit 0. |
| 18 | ATOM2_CH2_IRQ | ATOM2 channel 2 shared interrupt. See bit 0. |
| 17 | ATOM2_CH1_IRQ | ATOM2 channel 1 shared interrupt. See bit 0. |
| 16 | ATOM2_CH0_IRQ | ATOM2 channel 0 shared interrupt. See bit 0. |
| 15 | ATOM1_CH7_IRQ | ATOM1 channel 7 shared interrupt. See bit 0. |
| 14 | ATOM1_CH6_IRQ | ATOM1 channel 6 shared interrupt. See bit 0. |
| 13 | ATOM1_CH5_IRQ | ATOM1 channel 5 shared interrupt. See bit 0. |
| 12 | ATOM1_CH4_IRQ | ATOM1 channel 4 shared interrupt. See bit 0. |
| 11 | ATOM1_CH3_IRQ | ATOM1 channel 3 shared interrupt. See bit 0. |
| 10 | ATOM1_CH2_IRQ | ATOM1 channel 2 shared interrupt. See bit 0. |
| 9 | ATOM1_CH1_IRQ | ATOM1 channel 1 shared interrupt. See bit 0. |
| 8 | ATOM1_CH0_IRQ | ATOM1 channel 0 shared interrupt. See bit 0. |

**Table 25.180 GTM0ICMIRQG9 (ATOM Interrupt Group 0) Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | ATOM0_CH7_IRQ | ATOM0 channel 7 shared interrupt. See bit 0. |
| 6 | ATOM0_CH6_IRQ | ATOM0 channel 6 shared interrupt. See bit 0. |
| 5 | ATOM0_CH5_IRQ | ATOM0 channel 5 shared interrupt. See bit 0. |
| 4 | ATOM0_CH4_IRQ | ATOM0 channel 4 shared interrupt. See bit 0. |
| 3 | ATOM0_CH3_IRQ | ATOM0 channel 3 shared interrupt. See bit 0. |
| 2 | ATOM0_CH2_IRQ | ATOM0 channel 2 shared interrupt. See bit 0. |
| 1 | ATOM0_CH1_IRQ | ATOM0 channel 1 shared interrupt. See bit 0. |
| 0 | ATOM0_CH0_IRQ | ATOM0 channel 0 shared interrupt.<br>0: No interrupt occurred.<br>1: Interrupt was raised by the corresponding submodule.<br>**NOTE**<br>This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. |

### 25.15.5.5 GTM0ICMIRQGMEI (Module Error Interrupt)

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00630_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CMP_EIRQ | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | MCS1_EIRQ | MCS0_EIRQ | — | — | — | — | — | — | TIM1_EIRQ | TIM0_EIRQ | — | — | — | GTM_EIRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.181 GTM0ICMIRQGMEI (Module Error Interrupt) Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 25 | Reserved | These bits are always read as 0. |
| 24 | CMP_EIRQ | CMP error interrupt. See bit 0. |
| 23 to 14 | Reserved | These bits are always read as 0. |
| 13 | MCS1_EIRQ | MCS1 error interrupt. See bit 0. |
| 12 | MCS0_EIRQ | MCS0 error interrupt. See bit 0. |
| 11 to 6 | Reserved | These bits are always read as 0. |
| 5 | TIM1_EIRQ | TIM1 error interrupt. See bit 0. |
| 4 | TIM0_EIRQ | TIM0 error interrupt. See bit 0. |
| 3 to 1 | Reserved | These bits are always read as 0. |
| 0 | GTM_EIRQ | GTM Error interrupt request<br>0: No interrupt occurred.<br>1: Interrupt was raised by the corresponding submodule.<br>**NOTE**<br>This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding submodule. |

### 25.15.5.6 GTM0ICMIRQGCEI1 (Channel Error Interrupt 1)

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00638_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIM1_CH7_EIRQ | TIM1_CH6_EIRQ | TIM1_CH5_EIRQ | TIM1_CH4_EIRQ | TIM1_CH3_EIRQ | TIM1_CH2_EIRQ | TIM1_CH1_EIRQ | TIM1_CH0_EIRQ | TIM0_CH7_EIRQ | TIM0_CH6_EIRQ | TIM0_CH5_EIRQ | TIM0_CH4_EIRQ | TIM0_CH3_EIRQ | TIM0_CH2_EIRQ | TIM0_CH1_EIRQ | TIM0_CH0_EIRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.182 GTM0ICMIRQGCEI1 (Channel Error Interrupt 1) Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. |
| 15 | TIM1_CH7_EIRQ | TIM1 channel 7 error interrupt. See bit 0. |
| 14 | TIM1_CH6_EIRQ | TIM1 channel 6 error interrupt. See bit 0. |
| 13 | TIM1_CH5_EIRQ | TIM1 channel 5 error interrupt. See bit 0. |
| 12 | TIM1_CH4_EIRQ | TIM1 channel 4 error interrupt. See bit 0. |
| 11 | TIM1_CH3_EIRQ | TIM1 channel 3 error interrupt. See bit 0. |
| 10 | TIM1_CH2_EIRQ | TIM1 channel 2 error interrupt. See bit 0. |
| 9 | TIM1_CH1_EIRQ | TIM1 channel 1 error interrupt. See bit 0. |
| 8 | TIM1_CH0_EIRQ | TIM1 channel 0 error interrupt. See bit 0. |
| 7 | TIM0_CH7_EIRQ | TIM0 channel 7 error interrupt. See bit 0. |
| 6 | TIM0_CH6_EIRQ | TIM0 channel 6 error interrupt. See bit 0. |
| 5 | TIM0_CH5_EIRQ | TIM0 channel 5 error interrupt. See bit 0. |
| 4 | TIM0_CH4_EIRQ | TIM0 channel 4 error interrupt. See bit 0. |
| 3 | TIM0_CH3_EIRQ | TIM0 channel 3 error interrupt. See bit 0. |
| 2 | TIM0_CH2_EIRQ | TIM0 channel 2 error interrupt. See bit 0. |
| 1 | TIM0_CH1_EIRQ | TIM0 channel 1 error interrupt. See bit 0. |

**Table 25.182 GTM0ICMIRQGCEI1 (Channel Error Interrupt 1) Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | TIM0_CH0_EIRQ | TIM0 channel 0 error interrupt.<br>0: No error interrupt occurred.<br>1: Error interrupt was raised by the corresponding submodule.<br>**NOTE**<br>This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding submodule. |

### 25.15.5.7 GTM0ICMIRQGCEI3 (Channel Error Interrupt 3)

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00640_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCS1_CH7_EIRQ | MCS1_CH6_EIRQ | MCS1_CH5_EIRQ | MCS1_CH4_EIRQ | MCS1_CH3_EIRQ | MCS1_CH2_EIRQ | MCS1_CH1_EIRQ | MCS1_CH0_EIRQ | MCS0_CH7_EIRQ | MCS0_CH6_EIRQ | MCS0_CH5_EIRQ | MCS0_CH4_EIRQ | MCS0_CH3_EIRQ | MCS0_CH2_EIRQ | MCS0_CH1_EIRQ | MCS0_CH0_EIRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.183 GTM0ICMIRQGCEI3 (Channel Error Interrupt 3) Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. |
| 15 | MCS1_CH7_EIRQ | MCS1 channel 7 error interrupt. See bit 0. |
| 14 | MCS1_CH6_EIRQ | MCS1 channel 6 error interrupt. See bit 0. |
| 13 | MCS1_CH5_EIRQ | MCS1 channel 5 error interrupt. See bit 0. |
| 12 | MCS1_CH4_EIRQ | MCS1 channel 4 error interrupt. See bit 0. |
| 11 | MCS1_CH3_EIRQ | MCS1 channel 3 error interrupt. See bit 0. |
| 10 | MCS1_CH2_EIRQ | MCS1 channel 2 error interrupt. See bit 0. |
| 9 | MCS1_CH1_EIRQ | MCS1 channel 1 error interrupt. See bit 0. |
| 8 | MCS1_CH0_EIRQ | MCS1 channel 0 error interrupt. See bit 0. |
| 7 | MCS0_CH7_EIRQ | MCS0 channel 7 error interrupt. See bit 0. |
| 6 | MCS0_CH6_EIRQ | MCS0 channel 6 error interrupt. See bit 0. |
| 5 | MCS0_CH5_EIRQ | MCS0 channel 5 error interrupt. See bit 0. |
| 4 | MCS0_CH4_EIRQ | MCS0 channel 4 error interrupt. See bit 0. |
| 3 | MCS0_CH3_EIRQ | MCS0 channel 3 error interrupt. See bit 0. |
| 2 | MCS0_CH2_EIRQ | MCS0 channel 2 error interrupt. See bit 0. |
| 1 | MCS0_CH1_EIRQ | MCS0 channel 1 error interrupt. See bit 0. |

**Table 25.183 GTM0ICMIRQGCEI3 (Channel Error Interrupt 3) Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | MCS0_CH0_EIRQ | MCS0 channel 0 error interrupt.<br>0: No error interrupt occurred.<br>1: Error interrupt was raised by the corresponding submodule.<br>**NOTE**<br>This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding submodule. |

### 25.15.5.8 GTM0ICMIRQGMCS0CI (MCS 0 Channel Interrupt: 0 up to 31)

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00648_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCS0_CH31_IRQ | MCS0_CH30_IRQ | MCS0_CH29_IRQ | MCS0_CH28_IRQ | MCS0_CH27_IRQ | MCS0_CH26_IRQ | MCS0_CH25_IRQ | MCS0_CH24_IRQ | MCS0_CH23_IRQ | MCS0_CH22_IRQ | MCS0_CH21_IRQ | MCS0_CH20_IRQ | MCS0_CH19_IRQ | MCS0_CH18_IRQ | MCS0_CH17_IRQ | MCS0_CH16_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCS0_CH15_IRQ | MCS0_CH14_IRQ | MCS0_CH13_IRQ | MCS0_CH12_IRQ | MCS0_CH11_IRQ | MCS0_CH10_IRQ | MCS0_CH9_IRQ | MCS0_CH8_IRQ | MCS0_CH7_IRQ | MCS0_CH6_IRQ | MCS0_CH5_IRQ | MCS0_CH4_IRQ | MCS0_CH3_IRQ | MCS0_CH2_IRQ | MCS0_CH1_IRQ | MCS0_CH0_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.184 GTM0ICMIRQGMCS0CI (MCS 0 Channel Interrupt: 0 up to 31) Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | MCS0_CH31_IRQ | MCS0 channel 31 interrupt. See bit 0. |
| 30 | MCS0_CH30_IRQ | MCS0 channel 30 interrupt. See bit 0. |
| 29 | MCS0_CH29_IRQ | MCS0 channel 29 interrupt. See bit 0. |
| 28 | MCS0_CH28_IRQ | MCS0 channel 28 interrupt. See bit 0. |
| 27 | MCS0_CH27_IRQ | MCS0 channel 27 interrupt. See bit 0. |
| 26 | MCS0_CH26_IRQ | MCS0 channel 26 interrupt. See bit 0. |
| 25 | MCS0_CH25_IRQ | MCS0 channel 25 interrupt. See bit 0. |
| 24 | MCS0_CH24_IRQ | MCS0 channel 24 interrupt. See bit 0. |
| 23 | MCS0_CH23_IRQ | MCS0 channel 23 interrupt. See bit 0. |
| 22 | MCS0_CH22_IRQ | MCS0 channel 22 interrupt. See bit 0. |
| 21 | MCS0_CH21_IRQ | MCS0 channel 21 interrupt. See bit 0. |
| 20 | MCS0_CH20_IRQ | MCS0 channel 20 interrupt. See bit 0. |
| 19 | MCS0_CH19_IRQ | MCS0 channel 19 interrupt. See bit 0. |
| 18 | MCS0_CH18_IRQ | MCS0 channel 18 interrupt. See bit 0. |
| 17 | MCS0_CH17_IRQ | MCS0 channel 17 interrupt. See bit 0. |
| 16 | MCS0_CH16_IRQ | MCS0 channel 16 interrupt. See bit 0. |

Table 25.184 GTM0ICMIRQGMCS0CI (MCS 0 Channel Interrupt: 0 up to 31) Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | MCS0_CH15_IRQ | MCS0 channel 15 interrupt. See bit 0. |
| 14 | MCS0_CH14_IRQ | MCS0 channel 14 interrupt. See bit 0. |
| 13 | MCS0_CH13_IRQ | MCS0 channel 13 interrupt. See bit 0. |
| 12 | MCS0_CH12_IRQ | MCS0 channel 12 interrupt. See bit 0. |
| 11 | MCS0_CH11_IRQ | MCS0 channel 11 interrupt. See bit 0. |
| 10 | MCS0_CH10_IRQ | MCS0 channel 10 interrupt. See bit 0. |
| 9 | MCS0_CH9_IRQ | MCS0 channel 9 interrupt. See bit 0. |
| 8 | MCS0_CH8_IRQ | MCS0 channel 8 interrupt. See bit 0. |
| 7 | MCS0_CH7_IRQ | MCS0 channel 7 interrupt. See bit 0. |
| 6 | MCS0_CH6_IRQ | MCS0 channel 6 interrupt. See bit 0. |
| 5 | MCS0_CH5_IRQ | MCS0 channel 5 interrupt. See bit 0. |
| 4 | MCS0_CH4_IRQ | MCS0 channel 4 interrupt. See bit 0. |
| 3 | MCS0_CH3_IRQ | MCS0 channel 3 interrupt. See bit 0. |
| 2 | MCS0_CH2_IRQ | MCS0 channel 2 interrupt. See bit 0. |
| 1 | MCS0_CH1_IRQ | MCS0 channel 1 interrupt. See bit 0. |
| 0 | MCS0_CH0_IRQ | MCS0 channel 0 interrupt.<br>0: No interrupt occurred.<br>1: Interrupt was raised by the corresponding submodule.<br>**NOTE**<br>This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. |

### 25.15.5.9 GTM0ICMIRQGMCS0CEI (MCS 0 Channel Error Interrupt: 0 up to 31)

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00664_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCS0_CH31_EIRQ | MCS0_CH30_EIRQ | MCS0_CH29_EIRQ | MCS0_CH28_EIRQ | MCS0_CH27_EIRQ | MCS0_CH26_EIRQ | MCS0_CH25_EIRQ | MCS0_CH24_EIRQ | MCS0_CH23_EIRQ | MCS0_CH22_EIRQ | MCS0_CH21_EIRQ | MCS0_CH20_EIRQ | MCS0_CH19_EIRQ | MCS0_CH18_EIRQ | MCS0_CH17_EIRQ | MCS0_CH16_EIRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCS0_CH15_EIRQ | MCS0_CH14_EIRQ | MCS0_CH13_EIRQ | MCS0_CH12_EIRQ | MCS0_CH11_EIRQ | MCS0_CH10_EIRQ | MCS0_CH9_EIRQ | MCS0_CH8_EIRQ | MCS0_CH7_EIRQ | MCS0_CH6_EIRQ | MCS0_CH5_EIRQ | MCS0_CH4_EIRQ | MCS0_CH3_EIRQ | MCS0_CH2_EIRQ | MCS0_CH1_EIRQ | MCS0_CH0_EIRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.185 GTM0ICMIRQGMCS0CEI (MCS 0 Channel Error Interrupt: 0 up to 31) Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | MCS0_CH31_EIRQ | MCS0 channel 31 error interrupt. See bit 0. |
| 30 | MCS0_CH30_EIRQ | MCS0 channel 30 error interrupt. See bit 0. |
| 29 | MCS0_CH29_EIRQ | MCS0 channel 29 error interrupt. See bit 0. |
| 28 | MCS0_CH28_EIRQ | MCS0 channel 28 error interrupt. See bit 0. |
| 27 | MCS0_CH27_EIRQ | MCS0 channel 27 error interrupt. See bit 0. |
| 26 | MCS0_CH26_EIRQ | MCS0 channel 26 error interrupt. See bit 0. |
| 25 | MCS0_CH25_EIRQ | MCS0 channel 25 error interrupt. See bit 0. |
| 24 | MCS0_CH24_EIRQ | MCS0 channel 24 error interrupt. See bit 0. |
| 23 | MCS0_CH23_EIRQ | MCS0 channel 23 error interrupt. See bit 0. |
| 22 | MCS0_CH22_EIRQ | MCS0 channel 22 error interrupt. See bit 0. |
| 21 | MCS0_CH21_EIRQ | MCS0 channel 21 error interrupt. See bit 0. |
| 20 | MCS0_CH20_EIRQ | MCS0 channel 20 error interrupt. See bit 0. |
| 19 | MCS0_CH19_EIRQ | MCS0 channel 19 error interrupt. See bit 0. |
| 18 | MCS0_CH18_EIRQ | MCS0 channel 18 error interrupt. See bit 0. |
| 17 | MCS0_CH17_EIRQ | MCS0 channel 17 error interrupt. See bit 0. |
| 16 | MCS0_CH16_EIRQ | MCS0 channel 16 error interrupt. See bit 0. |

**Table 25.185 GTM0ICMIRQGMCS0CEI (MCS 0 Channel Error Interrupt: 0 up to 31) Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | MCS0_CH15_EIRQ | MCS0 channel 15 error interrupt. See bit 0. |
| 14 | MCS0_CH14_EIRQ | MCS0 channel 14 error interrupt. See bit 0. |
| 13 | MCS0_CH13_EIRQ | MCS0 channel 13 error interrupt. See bit 0. |
| 12 | MCS0_CH12_EIRQ | MCS0 channel 12 error interrupt. See bit 0. |
| 11 | MCS0_CH11_EIRQ | MCS0 channel 11 error interrupt. See bit 0. |
| 10 | MCS0_CH10_EIRQ | MCS0 channel 10 error interrupt. See bit 0. |
| 9 | MCS0_CH9_EIRQ | MCS0 channel 9 error interrupt. See bit 0. |
| 8 | MCS0_CH8_EIRQ | MCS0 channel 8 error interrupt. See bit 0. |
| 7 | MCS0_CH7_EIRQ | MCS0 channel 7 error interrupt. See bit 0. |
| 6 | MCS0_CH6_EIRQ | MCS0 channel 6 error interrupt. See bit 0. |
| 5 | MCS0_CH5_EIRQ | MCS0 channel 5 error interrupt. See bit 0. |
| 4 | MCS0_CH4_EIRQ | MCS0 channel 4 error interrupt. See bit 0. |
| 3 | MCS0_CH3_EIRQ | MCS0 channel 3 error interrupt. See bit 0. |
| 2 | MCS0_CH2_EIRQ | MCS0 channel 2 error interrupt. See bit 0. |
| 1 | MCS0_CH1_EIRQ | MCS0 channel 1 error interrupt. See bit 0. |
| 0 | MCS0_CH0_EIRQ | MCS0 channel 0 error interrupt.<br>0: No interrupt occurred.<br>1: Interrupt was raised by the corresponding submodule.<br>**NOTE**<br>This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. |

## 25.15.5.10GTM0ICMIRQGMCS1CI (MCS 1 Channel Interrupt: 0 up to 31)

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $0064C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCS1_CH31_IRQ | MCS1_CH30_IRQ | MCS1_CH29_IRQ | MCS1_CH28_IRQ | MCS1_CH27_IRQ | MCS1_CH26_IRQ | MCS1_CH25_IRQ | MCS1_CH24_IRQ | MCS1_CH23_IRQ | MCS1_CH22_IRQ | MCS1_CH21_IRQ | MCS1_CH20_IRQ | MCS1_CH19_IRQ | MCS1_CH18_IRQ | MCS1_CH17_IRQ | MCS1_CH16_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCS1_CH15_IRQ | MCS1_CH14_IRQ | MCS1_CH13_IRQ | MCS1_CH12_IRQ | MCS1_CH11_IRQ | MCS1_CH10_IRQ | MCS1_CH9_IRQ | MCS1_CH8_IRQ | MCS1_CH7_IRQ | MCS1_CH6_IRQ | MCS1_CH5_IRQ | MCS1_CH4_IRQ | MCS1_CH3_IRQ | MCS1_CH2_IRQ | MCS1_CH1_IRQ | MCS1_CH0_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.186 GTM0ICMIRQGMCS1CI (MCS 1 Channel Interrupt: 0 up to 31) Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | MCS1_CH31_IRQ | MCS1 channel 31 interrupt. See bit 0. |
| 30 | MCS1_CH30_IRQ | MCS1 channel 30 interrupt. See bit 0. |
| 29 | MCS1_CH29_IRQ | MCS1 channel 29 interrupt. See bit 0. |
| 28 | MCS1_CH28_IRQ | MCS1 channel 28 interrupt. See bit 0. |
| 27 | MCS1_CH27_IRQ | MCS1 channel 27 interrupt. See bit 0. |
| 26 | MCS1_CH26_IRQ | MCS1 channel 26 interrupt. See bit 0. |
| 25 | MCS1_CH25_IRQ | MCS1 channel 25 interrupt. See bit 0. |
| 24 | MCS1_CH24_IRQ | MCS1 channel 24 interrupt. See bit 0. |
| 23 | MCS1_CH23_IRQ | MCS1 channel 23 interrupt. See bit 0. |
| 22 | MCS1_CH22_IRQ | MCS1 channel 22 interrupt. See bit 0. |
| 21 | MCS1_CH21_IRQ | MCS1 channel 21 interrupt. See bit 0. |
| 20 | MCS1_CH20_IRQ | MCS1 channel 20 interrupt. See bit 0. |
| 19 | MCS1_CH19_IRQ | MCS1 channel 19 interrupt. See bit 0. |
| 18 | MCS1_CH18_IRQ | MCS1 channel 18 interrupt. See bit 0. |
| 17 | MCS1_CH17_IRQ | MCS1 channel 17 interrupt. See bit 0. |
| 16 | MCS1_CH16_IRQ | MCS1 channel 16 interrupt. See bit 0. |

Table 25.186 GTM0ICMIRQGMCS1CI (MCS 1 Channel Interrupt: 0 up to 31) Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | MCS1_CH15_IRQ | MCS1 channel 15 interrupt. See bit 0. |
| 14 | MCS1_CH14_IRQ | MCS1 channel 14 interrupt. See bit 0. |
| 13 | MCS1_CH13_IRQ | MCS1 channel 13 interrupt. See bit 0. |
| 12 | MCS1_CH12_IRQ | MCS1 channel 12 interrupt. See bit 0. |
| 11 | MCS1_CH11_IRQ | MCS1 channel 11 interrupt. See bit 0. |
| 10 | MCS1_CH10_IRQ | MCS1 channel 10 interrupt. See bit 0. |
| 9 | MCS1_CH9_IRQ | MCS1 channel 9 interrupt. See bit 0. |
| 8 | MCS1_CH8_IRQ | MCS1 channel 8 interrupt. See bit 0. |
| 7 | MCS1_CH7_IRQ | MCS1 channel 7 interrupt. See bit 0. |
| 6 | MCS1_CH6_IRQ | MCS1 channel 6 interrupt. See bit 0. |
| 5 | MCS1_CH5_IRQ | MCS1 channel 5 interrupt. See bit 0. |
| 4 | MCS1_CH4_IRQ | MCS1 channel 4 interrupt. See bit 0. |
| 3 | MCS1_CH3_IRQ | MCS1 channel 3 interrupt. See bit 0. |
| 2 | MCS1_CH2_IRQ | MCS1 channel 2 interrupt. See bit 0. |
| 1 | MCS1_CH1_IRQ | MCS1 channel 1 interrupt. See bit 0. |
| 0 | MCS1_CH0_IRQ | MCS1 channel 0 interrupt.<br>0: No interrupt occurred.<br>1: Interrupt was raised by the corresponding submodule.<br>**NOTE**<br>This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. |

## 25.15.5.11GTM0ICMIRQGMCS1CEI (MCS 1 Channel Error Interrupt: 0 up to 31)

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00668_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCS1_CH31_EIRQ | MCS1_CH30_EIRQ | MCS1_CH29_EIRQ | MCS1_CH28_EIRQ | MCS1_CH27_EIRQ | MCS1_CH26_EIRQ | MCS1_CH25_EIRQ | MCS1_CH24_EIRQ | MCS1_CH23_EIRQ | MCS1_CH22_EIRQ | MCS1_CH21_EIRQ | MCS1_CH20_EIRQ | MCS1_CH19_EIRQ | MCS1_CH18_EIRQ | MCS1_CH17_EIRQ | MCS1_CH16_EIRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCS1_CH15_EIRQ | MCS1_CH14_EIRQ | MCS1_CH13_EIRQ | MCS1_CH12_EIRQ | MCS1_CH11_EIRQ | MCS1_CH10_EIRQ | MCS1_CH9_EIRQ | MCS1_CH8_EIRQ | MCS1_CH7_EIRQ | MCS1_CH6_EIRQ | MCS1_CH5_EIRQ | MCS1_CH4_EIRQ | MCS1_CH3_EIRQ | MCS1_CH2_EIRQ | MCS1_CH1_EIRQ | MCS1_CH0_EIRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.187 GTM0ICMIRQGMCS1CEI (MCS 1 Channel Error Interrupt: 0 up to 31) Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | MCS1_CH31_EIRQ | MCS1 channel 31 error interrupt. See bit 0. |
| 30 | MCS1_CH30_EIRQ | MCS1 channel 30 error interrupt. See bit 0. |
| 29 | MCS1_CH29_EIRQ | MCS1 channel 29 error interrupt. See bit 0. |
| 28 | MCS1_CH28_EIRQ | MCS1 channel 28 error interrupt. See bit 0. |
| 27 | MCS1_CH27_EIRQ | MCS1 channel 27 error interrupt. See bit 0. |
| 26 | MCS1_CH26_EIRQ | MCS1 channel 26 error interrupt. See bit 0. |
| 25 | MCS1_CH25_EIRQ | MCS1 channel 25 error interrupt. See bit 0. |
| 24 | MCS1_CH24_EIRQ | MCS1 channel 24 error interrupt. See bit 0. |
| 23 | MCS1_CH23_EIRQ | MCS1 channel 23 error interrupt. See bit 0. |
| 22 | MCS1_CH22_EIRQ | MCS1 channel 22 error interrupt. See bit 0. |
| 21 | MCS1_CH21_EIRQ | MCS1 channel 21 error interrupt. See bit 0. |
| 20 | MCS1_CH20_EIRQ | MCS1 channel 20 error interrupt. See bit 0. |
| 19 | MCS1_CH19_EIRQ | MCS1 channel 19 error interrupt. See bit 0. |
| 18 | MCS1_CH18_EIRQ | MCS1 channel 18 error interrupt. See bit 0. |
| 17 | MCS1_CH17_EIRQ | MCS1 channel 17 error interrupt. See bit 0. |
| 16 | MCS1_CH16_EIRQ | MCS1 channel 16 error interrupt. See bit 0. |

Table 25.187 GTM0ICMIRQGMCS1CEI (MCS 1 Channel Error Interrupt: 0 up to 31) Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | MCS1_CH15_EIRQ | MCS1 channel 15 error interrupt. See bit 0. |
| 14 | MCS1_CH14_EIRQ | MCS1 channel 14 error interrupt. See bit 0. |
| 13 | MCS1_CH13_EIRQ | MCS1 channel 13 error interrupt. See bit 0. |
| 12 | MCS1_CH12_EIRQ | MCS1 channel 12 error interrupt. See bit 0. |
| 11 | MCS1_CH11_EIRQ | MCS1 channel 11 error interrupt. See bit 0. |
| 10 | MCS1_CH10_EIRQ | MCS1 channel 10 error interrupt. See bit 0. |
| 9 | MCS1_CH9_EIRQ | MCS1 channel 9 error interrupt. See bit 0. |
| 8 | MCS1_CH8_EIRQ | MCS1 channel 8 error interrupt. See bit 0. |
| 7 | MCS1_CH7_EIRQ | MCS1 channel 7 error interrupt. See bit 0. |
| 6 | MCS1_CH6_EIRQ | MCS1 channel 6 error interrupt. See bit 0. |
| 5 | MCS1_CH5_EIRQ | MCS1 channel 5 error interrupt. See bit 0. |
| 4 | MCS1_CH4_EIRQ | MCS1 channel 4 error interrupt. See bit 0. |
| 3 | MCS1_CH3_EIRQ | MCS1 channel 3 error interrupt. See bit 0. |
| 2 | MCS1_CH2_EIRQ | MCS1 channel 2 error interrupt. See bit 0. |
| 1 | MCS1_CH1_EIRQ | MCS1 channel 1 error interrupt. See bit 0. |
| 0 | MCS1_CH0_EIRQ | MCS1 channel 0 error interrupt.<br>0: No interrupt occurred.<br>1: Interrupt was raised by the corresponding submodule.<br>**NOTE**<br>This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. |

# 25.16 Output Compare Unit (CMP)

## 25.16.1 Overview

The Output Compare Unit (CMP) is designed for the use in safety relevant applications. The main idea is to have the possibility to duplicate outputs in order to be compared in this unit. Because of the simple EXOR function used it is necessary to ensure the total cycle accurate output behavior of the output modules to be compared. This is given when two DTM units produce output signals at the same time stamp. It is not necessary to compare each output channel with each other.

The CMP enables the comparison of 20 channels of the DTM units respectively and is restricted to neighbor channels. Thus, channel 0 is compared with channel 1, channel 2 with 3 and so on until the comparison of channel 18 with channel 19.

### 25.16.1.1 Architecture of the Compare Unit

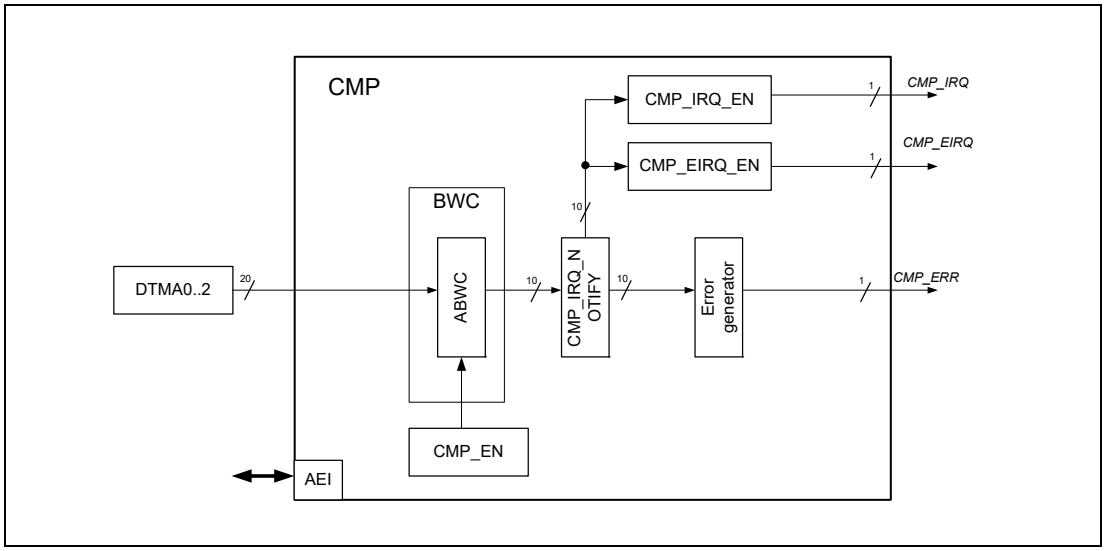



Figure 25.65 Architecture of the Compare Unit

## 25.16.2 Bitwise Compare Unit (BWC)

The Bitwise Compare Unit compares in pairs the combinations shown in following table

Table 25.188 Bitwise Compare Unit (BWC)

| ABWC Comparator Number | DTMA Bit Number one | DTMA Bit Number Two | Output Number |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 1 | 2 | 3 | 1 |
| 2 | 4 | 5 | 2 |
| 3 | 6 | 7 | 3 |
| 4 | 8 | 9 | 4 |
| 5 | 10 | 11 | 5 |
| 6 | 12 | 13 | 6 |
| 7 | 14 | 15 | 7 |
| 8 | 16 | 17 | 8 |
| 9 | 18 | 19 | 9 |

### 25.16.3 Configuration of the Compare Unit

Because of the restrictions described in the section above the Compare Unit consists of 10 antivalence (EXOR) elements, a select register GTM0CMPEN which selects the corresponding comparisons and a status register GTM0CMPIRQNOTIFY which shows and stores each mismatching result, when selected.

For each DTMA with an odd-channel DTM the MSB is not used for comparison to make sure of correct EXOR functionality.

For each with GTM0CMPIRQEN enabled mismatching error an interrupt signal on CMP_IRQ is generated.

For each with GTM0CMPIRQEN enabled mismatching error an interrupt signal on CMP_EIRQ is generated.

### 25.16.4 Error Generator

The error generator generates an error signal to be transmitted directly to the MON unit and independently from the CMP_IRQ and CMP_EIRQ. The error is set when in the GTM0CMPIRQNOTIFY register at least one bit is set. The GTM0CMPIRQNOTIFY bits are not mask able for this purpose.

The CMP_ERR output reflects its status in the status register of the Monitor Unit, which is to be polled by the CPU.

### 25.16.5 CMP Interrupt Signal

The CMP submodule has two interrupt signals, one normal interrupt and one error interrupt. The source of both interrupt can be determined by reading the GTM0CMPIRQNOTIFY register under consideration of GTM0CMPIRQEN register and GTM0CMPIRQEN register. Each source can be forced separately for debug purposes using the interrupt force GTM0CMPIRQFORCINT register. GTM0CMPIRQMODE configures interrupt output characteristic. All interrupt modes are described in detail in **Section 25.6.5, GTM-IP Interrupt Concept** .

**Table 25.189 CMP Interrupt Signal**

| Signal | Description |
|---|---|
| CMP_EIRQ | Mismatching interrupt of outputs to be compared, when enabled |
| CMP_IRQ | Mismatching interrupt of outputs to be compared, when enabled |

### 25.16.6 CMP Configuration Registers Overview

CMP contains following configuration registers:

**Table 25.190 Register list**

| Symbol | Register Name | Details in Section |
|---|---|---|
| GTM0CMPEN | Comparator enable register | **25.16.7.1** |
| GTM0CMPIRQNOTIFY | Event notification register | **25.16.7.2** |
| GTM0CMPIRQEN | Interrupt enable register | **25.16.7.3** |
| GTM0CMPIRQFORCINT | Interrupt force register | **25.16.7.4** |
| GTM0CMPIRQMODE | IRQ mode configuration register | **25.16.7.5** |
| GTM0CMPEIRQEN | Error interrupt enable register | **25.16.7.6** |

## 25.16.7 CMP Configuration Registers Description

### 25.16.7.1 GTM0CMPEN

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00200_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TBWC11_EN | TBWC10_EN | TBWC9_EN | TBWC8_EN | TBWC7_EN | TBWC6_EN | TBWC5_EN | TBWC4_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBWC3_EN | TBWC2_EN | TBWC1_EN | TBWC0_EN | ABWC11_EN | ABWC10_EN | ABWC9_EN | ABWC8_EN | ABWC7_EN | ABWC6_EN | ABWC5_EN | ABWC4_EN | ABWC3_EN | ABWC2_EN | ABWC1_EN | ABWC0_EN |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.191 GTM0CMPEN Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 | TBWC11_EN | Enable comparator 11 in TBWC (see **Section 25.16.2** ). See bit 12. |
| 22 | TBWC10_EN | Enable comparator 10 in TBWC (see **Section 25.16.2**). See bit 12. |
| 21 | TBWC9_EN | Enable comparator 9 in TBWC (see **Section 25.16.2)**. See bit 12. |
| 20 | TBWC8_EN | Enable comparator 8 in TBWC (see **Section 25.16.2**). See bit 12. |
| 19 | TBWC7_EN | Enable comparator 7 in TBWC (see **Section 25.16.2**). See bit 12. |
| 18 | TBWC6_EN | Enable comparator 6 in TBWC (see **Section 25.16.2**). See bit 12. |
| 17 | TBWC5_EN | Enable comparator 5 in TBWC (see **Section 25.16.2**). See bit 12. |
| 16 | TBWC4_EN | Enable comparator 4 in TBWC (see **Section 25.16.2**). See bit 12. |
| 15 | TBWC3_EN | Enable comparator 3 in TBWC (see **Section 25.16.2**). See bit 12. |
| 14 | TBWC2_EN | Enable comparator 2 in TBWC (see **Section 25.16.2**). See bit 12. |
| 13 | TBWC1_EN | Enable comparator 1 in TBWC (see **Section 25.16.2**). See bit 12. |
| 12 | TBWC0_EN | Enable comparator 0 in TBWC (see **Section 25.16.2**).<br>0: TBWC comparator 0 is disabled.<br>1: TBWC comparator 0 is enabled. |
| 11 | ABWC11_EN | Enable comparator 11 in ABWC (see **Section 25.16.2**). See bit 0. |
| 10 | ABWC10_EN | Enable comparator 10 in ABWC (see **Section 25.16.2**). See bit 0. |
| 9 | ABWC9_EN | Enable comparator 9 in ABWC (see **Section 25.16.2**). See bit 0. |
| 8 | ABWC8_EN | Enable comparator 8 in ABWC (see **Section 25.16.2**). See bit 0. |
| 7 | ABWC7_EN | Enable comparator 7 in ABWC (see **Section 25.16.2**). See bit 0. |
| 6 | ABWC6_EN | Enable comparator 6 in ABWC (see **Section 25.16.2**). See bit 0. |
| 5 | ABWC5_EN | Enable comparator 5 in ABWC (see **Section 25.16.2**). See bit 0. |
| 4 | ABWC4_EN | Enable comparator 4 in ABWC (see **Section 25.16.2**). See bit 0. |
| 3 | ABWC3_EN | Enable comparator 3 in ABWC (see **Section 25.16.2**). See bit 0. |
| 2 | ABWC2_EN | Enable comparator 2 in ABWC (see **Section 25.16.2**). See bit 0. |
| 1 | ABWC1_EN | Enable comparator 1 in ABWC (see **Section 25.16.2**). See bit 0. |

**Table 25.191 GTM0CMPEN Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | ABWC0_EN | Enable comparator 0 in ABWC (see **Section 25.16.2**).<br>0: ABWC Comparator 0 is disabled<br>1: ABWC Comparator 0 is enabled |

## 25.16.7.2 GTM0CMPIRQNOTIFY

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00204_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TBWC11 | TBWC10 | TBWC9 | TBWC8 | TBWC7 | TBWC6 | TBWC5 | TBWC4 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBWC3 | TBWC2 | TBWC1 | TBWC0 | ABWC11 | ABWC10 | ABWC9 | ABWC8 | ABWC7 | ABWC6 | ABWC5 | ABWC4 | ABWC3 | ABWC2 | ABWC1 | ABWC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 25.192 GTM0CMPIRQNOTIFY Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. |
| 23 | TBWC11 | TOM sub modules outputs bitwise comparator 11 error indication. See bit 12. |
| 22 | TBWC10 | TOM sub modules outputs bitwise comparator 10 error indication. See bit 12. |
| 21 | TBWC9 | TOM sub modules outputs bitwise comparator 9 error indication. See bit 12. |
| 20 | TBWC8 | TOM sub modules outputs bitwise comparator 8 error indication. See bit 12. |
| 19 | TBWC7 | TOM sub modules outputs bitwise comparator 7 error indication. See bit 12. |
| 18 | TBWC6 | TOM sub modules outputs bitwise comparator 6 error indication. See bit 12. |
| 17 | TBWC5 | TOM sub modules outputs bitwise comparator 5 error indication. See bit 12. |
| 16 | TBWC4 | TOM sub modules outputs bitwise comparator 4 error indication. See bit 12. |
| 15 | TBWC3 | TOM sub modules outputs bitwise comparator 3 error indication. See bit 12. |
| 14 | TBWC2 | TOM sub modules outputs bitwise comparator 2 error indication. See bit 12. |
| 13 | TBWC1 | TOM sub modules outputs bitwise comparator 1 error indication. See bit 12. |
| 12 | TBWC0 | TOM sub modules outputs bitwise comparator 0 error indication.<br>0: No error recognized on TOM sub modules bits 0 and 1 (see **Section 25.16.2)** .<br>1: An error was recognized on corresponding TOM sub modules bits.<br><br>0: No error recognized on DTMT sub modules bits 0 and 1 (see **Section 25.16.2**).<br>1: An error was recognized on corresponding DTMT sub modules bits.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

**Table 25.192 GTM0CMPIRQNOTIFY Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | ABWC11 | Error indication for ABWC11. See bit 0. |
| 10 | ABWC10 | Error indication for ABWC10. See bit 0. |
| 9 | ABWC9 | Error indication for ABWC9. See bit 0. |
| 8 | ABWC8 | Error indication for ABWC8. See bit 0. |
| 7 | ABWC7 | Error indication for ABWC7. See bit 0. |
| 6 | ABWC6 | Error indication for ABWC6. See bit 0. |
| 5 | ABWC5 | Error indication for ABWC5. See bit 0. |
| 4 | ABWC4 | Error indication for ABWC4. See bit 0. |
| 3 | ABWC3 | Error indication for ABWC3. See bit 0. |
| 2 | ABWC2 | Error indication for ABWC2. See bit 0. |
| 1 | ABWC1 | Error indication for ABWC1. See bit 0. |
| 0 | ABWC0 | Error indication for ABWC0.<br>0: No error recognized on DTMA sub modules bits 0 and 1 (see **Section 25.16.2**).<br>1: An error was recognized on corresponding DTMA sub modules bits.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

### 25.16.7.3 GTM0CMPIRQEN

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00208_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TBWC11_EN_IRQ | TBWC10_EN_IRQ | TBWC9_EN_IRQ | TBWC8_EN_IRQ | TBWC7_EN_IRQ | TBWC6_EN_IRQ | TBWC5_EN_IRQ | TBWC4_EN_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBWC3_EN_IRQ | TBWC2_EN_IRQ | TBWC1_EN_IRQ | TBWC0_EN_IRQ | ABWC11_EN_IRQ | ABWC10_EN_IRQ | ABWC9_EN_IRQ | ABWC8_EN_IRQ | ABWC7_EN_IRQ | ABWC6_EN_IRQ | ABWC5_EN_IRQ | ABWC4_EN_IRQ | ABWC3_EN_IRQ | ABWC2_EN_IRQ | ABWC1_EN_IRQ | ABWC0_EN_IRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.193 GTM0CMPIRQEN Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 | TBWC11_EN_IRQ | Enable TBWC11 interrupt source for CMP_IRQ line. See bit 12. |
| 22 | TBWC10_EN_IRQ | Enable TBWC10 interrupt source for CMP_IRQ line. See bit 12. |
| 21 | TBWC9_EN_IRQ | Enable TBWC9 interrupt source for CMP_IRQ line. See bit 12. |
| 20 | TBWC8_EN_IRQ | Enable TBWC8 interrupt source for CMP_IRQ line. See bit 12. |
| 19 | TBWC7_EN_IRQ | Enable TBWC7 interrupt source for CMP_IRQ line. See bit 12. |
| 18 | TBWC6_EN_IRQ | Enable TBWC6 interrupt source for CMP_IRQ line. See bit 12. |
| 17 | TBWC5_EN_IRQ | Enable TBWC5 interrupt source for CMP_IRQ line. See bit 12. |
| 16 | TBWC4_EN_IRQ | Enable TBWC4 interrupt source for CMP_IRQ line. See bit 12. |
| 15 | TBWC3_EN_IRQ | Enable TBWC3 interrupt source for CMP_IRQ line. See bit 12. |
| 14 | TBWC2_EN_IRQ | Enable TBWC2 interrupt source for CMP_IRQ line. See bit 12. |
| 13 | TBWC1_EN_IRQ | Enable TBWC1 interrupt source for CMP_IRQ line. See bit 12. |
| 12 | TBWC0_EN_IRQ | Enable TBWC0 interrupt source for CMP_IRQ line.<br>0: Interrupt source TBWC0 is disabled.<br>1: Interrupt source TBWC0 is enabled. |
| 11 | ABWC11_EN_IRQ | Enable ABWC11 interrupt source for CMP_IRQ line. See bit 0. |
| 10 | ABWC10_EN_IRQ | Enable ABWC10 interrupt source for CMP_IRQ line. See bit 0. |
| 9 | ABWC9_EN_IRQ | Enable ABWC9 interrupt source for CMP_IRQ line. See bit 0. |
| 8 | ABWC8_EN_IRQ | Enable ABWC8 interrupt source for CMP_IRQ line. See bit 0. |

**Table 25.193 GTM0CMPIRQEN Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | ABWC7_EN_IRQ | Enable ABWC7 interrupt source for CMP_IRQ line. See bit 0. |
| 6 | ABWC6_EN_IRQ | Enable ABWC6 interrupt source for CMP_IRQ line. See bit 0. |
| 5 | ABWC5_EN_IRQ | Enable ABWC5 interrupt source for CMP_IRQ line. See bit 0. |
| 4 | ABWC4_EN_IRQ | Enable ABWC4 interrupt source for CMP_IRQ line. See bit 0. |
| 3 | ABWC3_EN_IRQ | Enable ABWC3 interrupt source for CMP_IRQ line. See bit 0. |
| 2 | ABWC2_EN_IRQ | Enable ABWC2 interrupt source for CMP_IRQ line. See bit 0. |
| 1 | ABWC1_EN_IRQ | Enable ABWC1 interrupt source for CMP_IRQ line. See bit 0. |
| 0 | ABWC0_EN_IRQ | Enable ABWC0 interrupt source for CMP_IRQ line.<br>0: Interrupt source ABWC0 is disabled.<br>1: Interrupt source ABWC0 is enabled. |

### 25.16.7.4 GTM0CMPIRQFORCINT

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $0020C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TRG_TBWC11 | TRG_TBWC10 | TRG_TBWC9 | TRG_TBWC8 | TRG_TBWC7 | TRG_TBWC6 | TRG_TBWC5 | TRG_TBWC4 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRG_TBWC3 | TRG_TBWC2 | TRG_TBWC1 | TRG_TBWC0 | TRG_ABWC11 | TRG_ABWC10 | TRG_ABWC9 | TRG_ABWC8 | TRG_ABWC7 | TRG_ABWC6 | TRG_ABWC5 | TRG_ABWC4 | TRG_ABWC3 | TRG_ABWC2 | TRG_ABWC1 | TRG_ABWC0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.194 GTM0CMPIRQFORCINT Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 | TRG_TBWC11 | Trigger TBWC11 bit in GTM0CMPIRQNOTIFY register by software. See bit 12. |
| 22 | TRG_TBWC10 | Trigger TBWC10 bit in GTM0CMPIRQNOTIFY register by software. See bit 12. |
| 21 | TRG_TBWC9 | Trigger TBWC9 bit in GTM0CMPIRQNOTIFY register by software. See bit 12. |
| 20 | TRG_TBWC8 | Trigger TBWC8 bit in GTM0CMPIRQNOTIFY register by software. See bit 12. |
| 19 | TRG_TBWC7 | Trigger TBWC7 bit in GTM0CMPIRQNOTIFY register by software. See bit 12. |
| 18 | TRG_TBWC6 | Trigger TBWC6 bit in GTM0CMPIRQNOTIFY register by software. See bit 12. |
| 17 | TRG_TBWC5 | Trigger TBWC5 bit in GTM0CMPIRQNOTIFY register by software. See bit 12. |
| 16 | TRG_TBWC4 | Trigger TBWC4 bit in GTM0CMPIRQNOTIFY register by software. See bit 12. |
| 15 | TRG_TBWC3 | Trigger TBWC3 bit in GTM0CMPIRQNOTIFY register by software. See bit 12. |
| 14 | TRG_TBWC2 | Trigger TBWC2 bit in GTM0CMPIRQNOTIFY register by software. See bit 12. |
| 13 | TRG_TBWC1 | Trigger TBWC1 bit in GTM0CMPIRQNOTIFY register by software. See bit 12. |
| 12 | TRG_TBWC0 | Enable TBWC0 interrupt source for CMP_IRQ line.<br>0: Interrupt source TBWC0 is disabled.<br>1: Interrupt source TBWC0 is enabled. |
| 11 | TRG_ABWC11 | Trigger TBWC0 bit in GTM0CMPIRQNOTIFY register by software.<br>0: No event triggering.<br>1: Assert corresponding field in GTM0CMPIRQNOTIFY register. |
| 10 | TRG_ABWC10 | Trigger ABWC10 bit in GTM0CMPIRQNOTIFY register by software. See bit 0. |
| 9 | TRG_ABWC9 | Trigger ABWC9 bit in GTM0CMPIRQNOTIFY register by software. See bit 0. |
| 8 | TRG_ABWC8 | Trigger ABWC8 bit in GTM0CMPIRQNOTIFY register by software. See bit 0. |
| 7 | TRG_ABWC7 | Trigger ABWC7 bit in GTM0CMPIRQNOTIFY register by software. See bit 0. |
| 6 | TRG_ABWC6 | Trigger ABWC6 bit in GTM0CMPIRQNOTIFY register by software. See bit 0. |
| 5 | TRG_ABWC5 | Trigger ABWC5 bit in GTM0CMPIRQNOTIFY register by software. See bit 0. |
| 4 | TRG_ABWC4 | Trigger ABWC4 bit in GTM0CMPIRQNOTIFY register by software. See bit 0. |
| 3 | TRG_ABWC3 | Trigger ABWC3 bit in GTM0CMPIRQNOTIFY register by software. See bit 0. |
| 2 | TRG_ABWC2 | Trigger ABWC2 bit in GTM0CMPIRQNOTIFY register by software. See bit 0. |
| 1 | TRG_ABWC1 | Trigger ABWC1 bit in GTM0CMPIRQNOTIFY register by software. See bit 0. |

**Table 25.194 GTM0CMPIRQFORCINT Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | TRG_ABWC0 | Trigger ABWC0 bit in GTM0CMPIRQNOTIFY register by software.<br>0: No event triggering.<br>1: Assert corresponding field in GTM0CMPIRQNOTIFY register.<br>**NOTES**<br>1. This bit is cleared automatically after write.<br>2. This bit is write protected by bit RF_PROT of register GTM0GTMCTRL. |

## 25.16.7.5 GTM0CMPIRQMODE

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00210_H$

**Value after reset:** $0000\ 000X_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IRQ_MODE | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 25.195 GTM0CMPIRQMODE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1, 0 | IRQ_MODE | IRQ mode selection<br>00: Level mode<br>01: Pulse mode<br>10: Pulse-Notify mode<br>11: Single-Pulse mode<br>**NOTE**<br>The interrupt modes are described in **Section 25.6.5, GTM-IP Interrupt Concept** . |

### 25.16.7.6 GTM0CMPEIRQEN

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00208_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TBWC11_EN_EIRQ | TBWC10_EN_EIRQ | TBWC9_EN_EIRQ | TBWC8_EN_EIRQ | TBWC7_EN_EIRQ | TBWC6_EN_EIRQ | TBWC5_EN_EIRQ | TBWC4_EN_EIRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBWC3_EN_EIRQ | TBWC2_EN_EIRQ | TBWC1_EN_EIRQ | TBWC0_EN_EIRQ | ABWC11_EN_EIRQ | ABWC10_EN_EIRQ | ABWC9_EN_EIRQ | ABWC8_EN_EIRQ | ABWC7_EN_EIRQ | ABWC6_EN_EIRQ | ABWC5_EN_EIRQ | ABWC4_EN_EIRQ | ABWC3_EN_EIRQ | ABWC2_EN_EIRQ | ABWC1_EN_EIRQ | ABWC0_EN_EIRQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.196 GTM0CMPEIRQEN Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 | TBWC11_EN_EIRQ | Enable TBWC11 interrupt source for CMP_EIRQ line. See bit 12. |
| 22 | TBWC10_EN_EIRQ | Enable TBWC10 interrupt source for CMP_EIRQ line. See bit 12. |
| 21 | TBWC9_EN_EIRQ | Enable TBWC9 interrupt source for CMP_EIRQ line. See bit 12. |
| 20 | TBWC8_EN_EIRQ | Enable TBWC8 interrupt source for CMP_EIRQ line. See bit 12. |
| 19 | TBWC7_EN_EIRQ | Enable TBWC7 interrupt source for CMP_EIRQ line. See bit 12. |
| 18 | TBWC6_EN_EIRQ | Enable TBWC6 interrupt source for CMP_EIRQ line. See bit 12. |
| 17 | TBWC5_EN_EIRQ | Enable TBWC5 interrupt source for CMP_EIRQ line. See bit 12. |
| 16 | TBWC4_EN_EIRQ | Enable TBWC4 interrupt source for CMP_EIRQ line. See bit 12. |
| 15 | TBWC3_EN_EIRQ | Enable TBWC3 interrupt source for CMP_EIRQ line. See bit 12. |
| 14 | TBWC2_EN_EIRQ | Enable TBWC2 interrupt source for CMP_EIRQ line. See bit 12. |
| 13 | TBWC1_EN_EIRQ | Enable TBWC1 interrupt source for CMP_EIRQ line. See bit 12. |
| 12 | TBWC0_EN_EIRQ | Enable TBWC0 interrupt source for CMP_EIRQ line.<br>0: Interrupt source TBWC0 is disabled.<br>1: Interrupt source TBWC0 is enabled. |
| 11 | ABWC11_EN_EIRQ | Enable ABWC11 interrupt source for CMP_EIRQ line. See bit 0. |
| 10 | ABWC10_EN_EIRQ | Enable ABWC10 interrupt source for CMP_EIRQ line. See bit 0. |
| 9 | ABWC9_EN_EIRQ | Enable ABWC9 interrupt source for CMP_EIRQ line. See bit 0. |
| 8 | ABWC8_EN_EIRQ | Enable ABWC8 interrupt source for CMP_EIRQ line. See bit 0. |

**Table 25.196 GTM0CMPEIRQEN Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | ABWC7_EN_EIRQ | Enable ABWC7 interrupt source for CMP_EIRQ line. See bit 0. |
| 6 | ABWC6_EN_EIRQ | Enable ABWC6 interrupt source for CMP_EIRQ line. See bit 0. |
| 5 | ABWC5_EN_EIRQ | Enable ABWC5 interrupt source for CMP_EIRQ line. See bit 0. |
| 4 | ABWC4_EN_EIRQ | Enable ABWC4 interrupt source for CMP_EIRQ line. See bit 0. |
| 3 | ABWC3_EN_EIRQ | Enable ABWC3 interrupt source for CMP_EIRQ line. See bit 0. |
| 2 | ABWC2_EN_EIRQ | Enable ABWC2 interrupt source for CMP_EIRQ line. See bit 0. |
| 1 | ABWC1_EN_EIRQ | Enable ABWC1 interrupt source for CMP_EIRQ line. See bit 0. |
| 0 | ABWC0_EN_EIRQ | Enable ABWC0 interrupt source for CMP_EIRQ line<br>0: Interrupt source ABWC0 is disabled.<br>1: Interrupt source ABWC0 is enabled. |

# 25.17 Monitor Unit (MON)

## 25.17.1 Overview

The Monitor Unit (MON) is designed for the use in safety relevant applications. The main idea is to have a possibility to supervise common used circuitry and resources. In this way the activity of the clocks is supervised. In addition the characteristics of output signals can be checked in a MCS channel by a re-read-in via TIM and routing to the MCS. When the comparison fails an error signal is ge1nerated in MCS and sent to the monitor unit. One error signal per MCS summarizes the errors of all channels. By generating of an activity signal per channel for each such performed comparison, the activity of TIM, ARU and the used clocks is checked implicitly.

In addition the ARU cycle time could be also compared in a MCS channel to given values.

### 25.17.1.1 MON Block Diagram

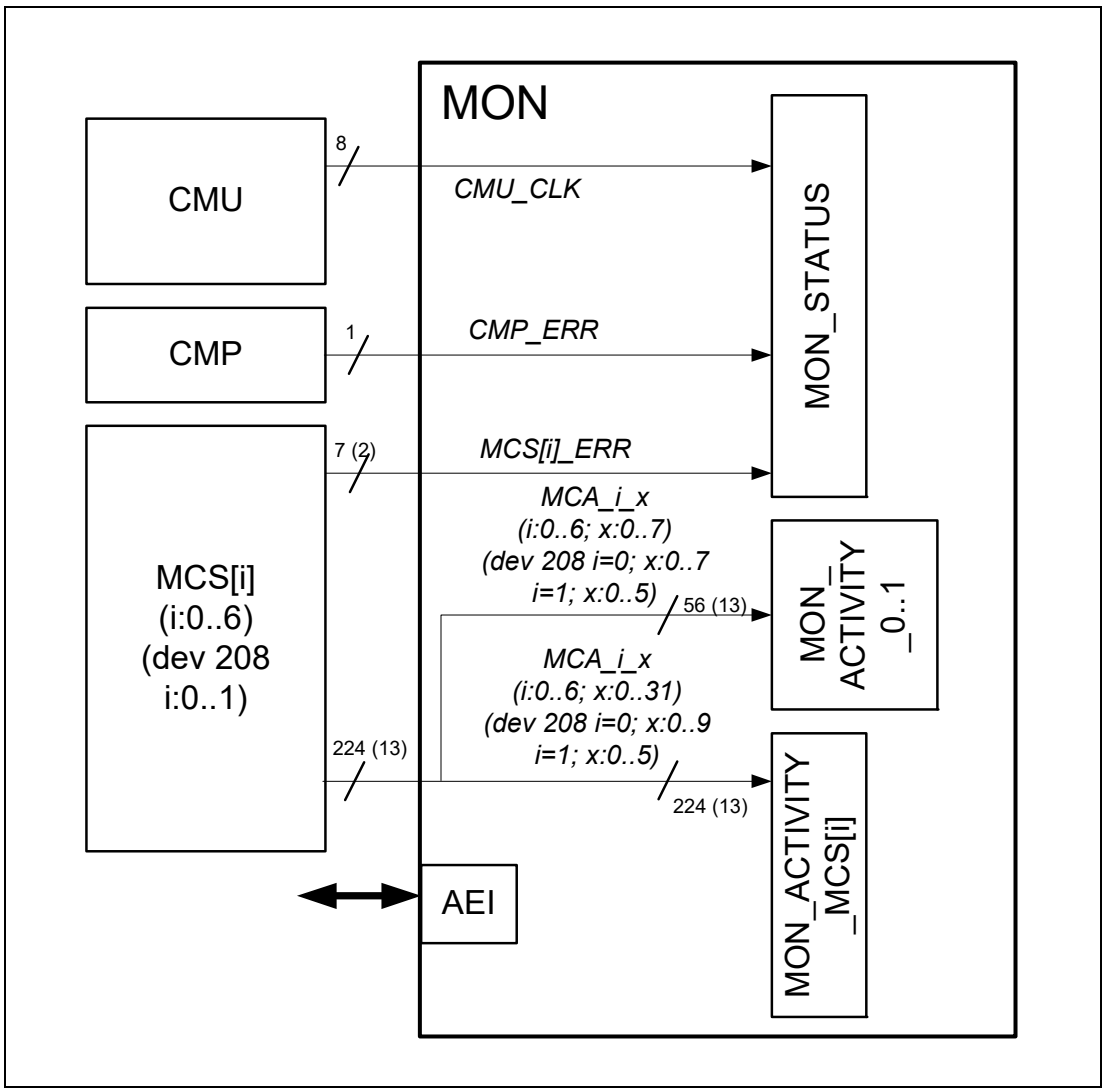



Figure 25.66 MON Block Diagram

#### 25.17.1.2 Realization without Activity Checker of the clock signals

An activity checker of the clock signals used is not needed because these signals are only enables to be used in combination with the system clock. Therefore the clock enables are to be checked to have a high value.

### 25.17.2 Clock Monitoring

The monitor unit has a connection to each of the 8 clocks CMU_CLK[x] (x = 0 to 7), provided by the CMU. Some of these clocks can be used for special tasks (see **Section 25.8, Clock Management Unit (CMU)** ).

The supervising of the clocks is done by scanning for activity of each clock.

A high value is defined as the state to be monitored.

When a high value of the clock enable is detected, the corresponding bit in the status register GTM0MONSTATUS is set.

The status register bits are reset by writing a one.

When the register is polled by the CPU and the time between two read accesses is higher than the period of the slowest clock, all bits of the corresponding clocks must have been set.

When polling in shorter time distances, not for all clocks an activity can be shown, although they are still working.

Because of the realization without a select register for the clock signals only the bits of the status register are to be considered for which the clock signal is enabled in the CMU.

### 25.17.3 CMP error Monitoring

The signal CMP_ERR is to be received directly from module CMP and is set if an error occurred.

### 25.17.4 Checking the Characteristics of Signals by MCS

By use of the MCS some given properties of signals can be checked. Such signals can be generated output signals of ATOM channels, which are re-read in into a TIM and the time stamp information is routed via ARU to the MCS module.

The corresponding MCS signal performs the check according to given properties. In this way signal high or low time as well as signal periods can be checked, also taking into account tolerances. When the check fails a MCS internal error signal is generated and ORed with the error signals of the other channels of the MCS module to an summarized error signal GTM0MCSiERR (i = 0 to 1).

For each MCS a summarized error signal is transmitted to MON and monitored in the GTM0MONSTATUS register.

In order to check the execution of the comparison for each MCS channel an activity signal is generated. In the MCA_x (x = 0 to 31) vector 32 bits for each MCS[i](i = 0 to 6) instance are combined. The activity signals are stored in the GTM0MONACTIVITYMCSi register. In addition the first 8 bits of MCS0 to 3 are stored in GTMMONACTIVITY0. The bits are set by a one signal and reset by writing a one to it (preferably after polling the status of the register).

For the first 8 MCS channels the activity information is stored twice (GTMMONACTIVITY0, GTM0MONACTIVITYMCSi). The same actitity bit is reset by writing one register only.

Because the activity signal shows the execution of a comparison, the involved units for providing the signals and execution of comparison (like TIM, ARU and MCS itself) are checked implicitly to work accordingly. Also the involved clocks and time bases are checked in this way.

### 25.17.5 Checking ARU Cycle Time

The cycle time of the ARU can be checked, when this is essential for safety purposes. This check can be performed by an MCS channel. It should be noted that the MCS program for measuring the ARU round trip time must add a tolerance value.

The resulting error is reported to the MON unit using the summarized error signal GTM0MCSiERR for each MCS module in addition to an interrupt, generated in MCS. The same signals and status bits are used as in the case of checking the signal characteristics.

The corresponding MCS is programmed to get a fixed data value at address 0x1FF. The data value is always zero and is not blocked. When getting the access the time stamp value TBU_TS0 is stored in a register. The next time getting the access the new TBU_TS0 value is stored and the difference between both values is compared with a given value. When the comparison fails, an error flag is set in the MCS internal status register, an interrupt is generated and the error signal GTM0MCSiERR is provided.

When the check is performed, an activity signal MCA_x (x = 0 to 31) is provided for each channel x for each MCS[i](i = 0 to 6) instance together with a summarized interrupt GTM0MCSiERR for each MCS.

The activity signal sets a bit in the GTMMONACTIVITY0 register.

The bits in the GTMMONACTIVITY0 registers are reset by writing a one.

When the check fails, an interrupt is generated and the error signal GTM0MCSiERR is provided for the MON unit.

**Figure 25.66** shows the block diagram of the Monitor Unit.

### 25.17.6 MON Interrupt Signals

The MON submodule has no interrupt signals.

### 25.17.7 MON Registers Overview

Following configuration registers are considered in MON sub module

**Table 25.197 Register list**

| Symbol | Register Name | Details in Section |
|---|---|---|
| GTM0MONSTATUS | Monitor Status register | **25.17.8.1** |
| GTMMONACTIVITY0 | Monitor activity register 0 | **25.17.8.2** |
| GTM0MONACTIVITYMCSi | Monitor activity register for MCS [z] (z = 0 to 6) | **25.17.8.3** |

## 25.17.8 MON Configuration Registers Description

### 25.17.8.1 GTM0MONSTATUS

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00180_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | MCS6_ERR | MCS5_ERR | MCS4_ERR | MCS3_ERR | MCS2_ERR | MCS1_ERR | MCS0_ERR | — | — | — | CMP_ERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ACT_CMU7 | ACT_CMU6 | ACT_CMU5 | ACT_CMU4 | ACT_CMU3 | ACT_CMU2 | ACT_CMU1 | ACT_CMU0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.198 GTM0MONSTATUS Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 26 | MCS6_ERR | Error detected at MCS6<br>**NOTE**<br>This bit will be readable only. |
| 25 | MCS5_ERR | Error detected at MCS5<br>**NOTE**<br>This bit will be readable only. |
| 24 | MCS4_ERR | Error detected at MCS4<br>**NOTE**<br>This bit will be readable only. |
| 23 | MCS3_ERR | Error detected at MCS3<br>**NOTE**<br>This bit will be readable only. |
| 22 | MCS2_ERR | Error detected at MCS2<br>**NOTE**<br>This bit will be readable only. |
| 21 | MCS1_ERR | Error detected at MCS1<br>**NOTE**<br>This bit will be readable only. |

**Table 25.198 GTM0MONSTATUS Register Contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 20 | MCS0_ERR | Error detected at MCS0<br>**NOTE**<br>This bit will be readable only. |
| 19 to 17 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 16 | CMP_ERR | Error detected at CMP<br>**NOTE**<br>This bit will be readable only. |
| 15 to 8 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 7 | ACT_CMU7 | CMU_CLK7 activity<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 6 | ACT_CMU6 | CMU_CLK6 activity<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 5 | ACT_CMU5 | CMU_CLK5 activity<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 4 | ACT_CMU4 | CMU_CLK4 activity<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 3 | ACT_CMU3 | CMU_CLK3 activity<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 2 | ACT_CMU2 | CMU_CLK2 activity<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 1 | ACT_CMU1 | CMU_CLK1 activity<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

**Table 25.198 GTM0MONSTATUS Register Contents (3/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 0 | ACT_CMU0 | CMU_CLK0 activity<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

**NOTES**

1. Bits16 and 20 to 26 are set, when the corresponding unit reports an error.
2. Bits 0 to 7 are set, when a high low slope is detected at the considered clock.
3. The MCS can be programmed to generate an error, when the comparison of signal values (duty time, cycle time) fails or also when the cycle time of the ARU (checking of the TBU_TS0 between two periodic accesses) is out of the expected range.

### 25.17.8.2 GTMMONACTIVITY0

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00184_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCA_3_7 | MCA_3_6 | MCA_3_5 | MCA_3_4 | MCA_3_3 | MCA_3_2 | MCA_3_1 | MCA_3_0 | MCA_2_7 | MCA_2_6 | MCA_2_5 | MCA_2_4 | MCA_2_3 | MCA_2_2 | MCA_2_1 | MCA_2_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCA_1_7 | MCA_1_6 | MCA_1_5 | MCA_1_4 | MCA_1_3 | MCA_1_2 | MCA_1_1 | MCA_1_0 | MCA_0_7 | MCA_0_6 | MCA_0_5 | MCA_0_4 | MCA_0_3 | MCA_0_2 | MCA_0_1 | MCA_0_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.199 GTMMONACTIVITY0 Register Contents (1/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | MCA_3_7 | Activity of check performed in module MCS 3 at channel 7.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 30 | MCA_3_6 | Activity of check performed in module MCS 3 at channel 6.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 29 | MCA_3_5 | Activity of check performed in module MCS 3 at channel 5.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 28 | MCA_3_4 | Activity of check performed in module MCS 3 at channel 4.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 27 | MCA_3_3 | Activity of check performed in module MCS 3 at channel 3.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 26 | MCA_3_2 | Activity of check performed in module MCS 3 at channel 2.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

**Table 25.199 GTMMONACTIVITY0 Register Contents (2/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 25 | MCA_3_1 | Activity of check performed in module MCS 3 at channel 1.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 24 | MCA_3_0 | Activity of check performed in module MCS 3 at channel 0.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 23 | MCA_2_7 | Activity of check performed in module MCS 2 at channel 7.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 22 | MCA_2_6 | Activity of check performed in module MCS 2 at channel 6.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 21 | MCA_2_5 | Activity of check performed in module MCS 2 at channel 5.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 20 | MCA_2_4 | Activity of check performed in module MCS 2 at channel 4.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 19 | MCA_2_3 | Activity of check performed in module MCS 2 at channel 3.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 18 | MCA_2_2 | Activity of check performed in module MCS 2 at channel 2.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 17 | MCA_2_1 | Activity of check performed in module MCS 2 at channel 1.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 16 | MCA_2_0 | Activity of check performed in module MCS 2 at channel 0.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

**Table 25.199 GTMMONACTIVITY0 Register Contents (3/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | MCA_1_7 | Activity of check performed in module MCS 1 at channel 7.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 14 | MCA_1_6 | Activity of check performed in module MCS 1 at channel 6.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 13 | MCA_1_5 | Activity of check performed in module MCS 1 at channel 5.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 12 | MCA_1_4 | Activity of check performed in module MCS 1 at channel 4.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 11 | MCA_1_3 | Activity of check performed in module MCS 1 at channel 3.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 10 | MCA_1_2 | Activity of check performed in module MCS 1 at channel 2.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 9 | MCA_1_1 | Activity of check performed in module MCS 1 at channel 1.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 8 | MCA_1_0 | Activity of check performed in module MCS 1 at channel 0.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 7 | MCA_0_7 | Activity of check performed in module MCS 0 at channel 7.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 6 | MCA_0_6 | Activity of check performed in module MCS 0 at channel 6.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

**Table 25.199 GTMMONACTIVITY0 Register Contents (4/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 5 | MCA_0_5 | Activity of check performed in module MCS 0 at channel 5.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 4 | MCA_0_4 | Activity of check performed in module MCS 0 at channel 4.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 3 | MCA_0_3 | Activity of check performed in module MCS 0 at channel 3.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 2 | MCA_0_2 | Activity of check performed in module MCS 0 at channel 2.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 1 | MCA_0_1 | Activity of check performed in module MCS 0 at channel 1.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 0 | MCA_0_0 | Activity of check performed in module MCS 0 at channel 0.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

**NOTE**

When not all MCS modules are implemented or the channels are not used for check purposes with supervising, the corresponding activity bits remain zero.

### 25.17.8.3 GTM0MONACTIVITYMCSi (i = 0, 1)

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0MONACTIVITYMCS0: <GTM_base> + $0018C_H$
GTM0MONACTIVITYMCS1: <GTM_base> + $00190_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCA_31 | MCA_30 | MCA_29 | MCA_28 | MCA_27 | MCA_26 | MCA_25 | MCA_24 | MCA_23 | MCA_22 | MCA_21 | MCA_20 | MCA_19 | MCA_18 | MCA_17 | MCA_16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MCA_15 | MCA_14 | MCA_13 | MCA_12 | MCA_11 | MCA_10 | MCA_9 | MCA_8 | MCA_7 | MCA_6 | MCA_5 | MCA_4 | MCA_3 | MCA_2 | MCA_1 | MCA_0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 25.200 GTM0MONACTIVITYMCSi Register Contents (1/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | MCA_31 | Activity of check performed in module MCS[i] at channel 31.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value ‘1’. A read access leaves the bit unchanged. |
| 30 | MCA_30 | Activity of check performed in module MCS[i] at channel 30.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value ‘1’. A read access leaves the bit unchanged. |
| 29 | MCA_29 | Activity of check performed in module MCS[i] at channel 29.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value ‘1’. A read access leaves the bit unchanged. |
| 28 | MCA_28 | Activity of check performed in module MCS[i] at channel 28.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value ‘1’. A read access leaves the bit unchanged. |
| 27 | MCA_27 | Activity of check performed in module MCS[i] at channel 27.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value ‘1’. A read access leaves the bit unchanged. |
| 26 | MCA_26 | Activity of check performed in module MCS[i] at channel 26.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value ‘1’. A read access leaves the bit unchanged. |

**Table 25.200 GTM0MONACTIVITYMCSi Register Contents (2/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 25 | MCA_25 | Activity of check performed in module MCS[i] at channel 25.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 24 | MCA_24 | Activity of check performed in module MCS[i] at channel 24.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 23 | MCA_23 | Activity of check performed in module MCS[i] at channel 23.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 22 | MCA_22 | Activity of check performed in module MCS[i] at channel 22.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 21 | MCA_21 | Activity of check performed in module MCS[i] at channel 21.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 20 | MCA_20 | Activity of check performed in module MCS[i] at channel 20.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 19 | MCA_19 | Activity of check performed in module MCS[i] at channel 19.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 18 | MCA_18 | Activity of check performed in module MCS[i] at channel 18.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 17 | MCA_17 | Activity of check performed in module MCS[i] at channel 17.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 16 | MCA_16 | Activity of check performed in module MCS[i] at channel 16.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

**Table 25.200 GTM0MONACTIVITYMCSi Register Contents (3/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 | MCA_15 | Activity of check performed in module MCS[i] at channel 15.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 14 | MCA_14 | Activity of check performed in module MCS[i] at channel 14.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 13 | MCA_13 | Activity of check performed in module MCS[i] at channel 13.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 12 | MCA_12 | Activity of check performed in module MCS[i] at channel 12.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 11 | MCA_11 | Activity of check performed in module MCS[i] at channel 11.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 10 | MCA_10 | Activity of check performed in module MCS[i] at channel 10.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 9 | MCA_9 | Activity of check performed in module MCS[i] at channel 9.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 8 | MCA_8 | Activity of check performed in module MCS[i] at channel 8.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 7 | MCA_7 | Activity of check performed in module MCS[i] at channel 7.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 6 | MCA_6 | Activity of check performed in module MCS[i] at channel 6.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

**Table 25.200 GTM0MONACTIVITYMCSi Register Contents (4/4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 5 | MCA_5 | Activity of check performed in module MCS[i] at channel 5.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 4 | MCA_4 | Activity of check performed in module MCS[i] at channel 4.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 3 | MCA_3 | Activity of check performed in module MCS[i] at channel 3.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 2 | MCA_2 | Activity of check performed in module MCS[i] at channel 2.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 1 | MCA_1 | Activity of check performed in module MCS[i] at channel 1.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |
| 0 | MCA_0 | Activity of check performed in module MCS[i] at channel 0.<br>**NOTE**<br>This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. |

**NOTES**

1. Unused MCA bits are reserved.
2. Read as zero should be written as zero.

# 25.18 GTM Device 207

## 25.18.1 Architecture Block Diagram

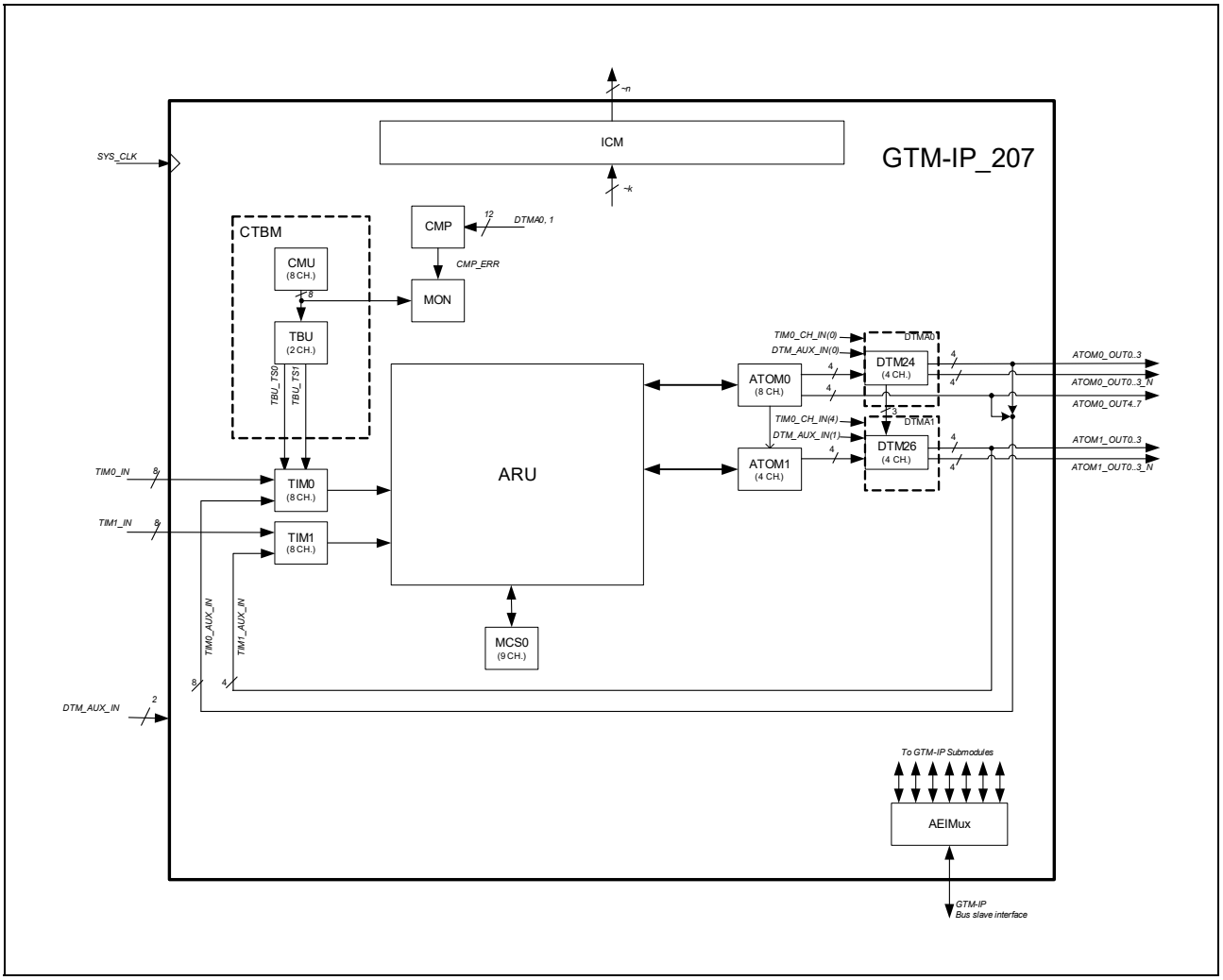



**Figure 25.67 Architecture Block Diagram**

### 25.18.2 GTM Device 207 Configuration

The following table lists the device configuration of device GTM-IP_207

**Table 25.201 GTM Device 207 Configuration**

| | Instances | Numbering | Channel |
|---|---|---|---|
| ARU | 1 | | |
| BRC | 0 | | |
| PSM | 0 | | |
| CMU | 1 | — | 8 |
| TBU | 1 | — | 2 |
| TIM | 2 | 0, 1 | 8, 8 |
| TOM | 0 | | |
| ATOM | 2 | 0, 1 | 8, 4 |
| DTM | 2 | 24, 26 | 4, 4 |
| MCS | 1 | 0 | 9 |
| MCFG | 0 | — | — |
| MAP | 0 | | |
| DPLL | 0 | | |
| SPE | 0 | | |
| ICM | 1 | — | — |
| CMP | 1 | — | — |
| MON | 1 | — | — |

Following configuration constants referenced in specification are defined for this device:

**Table 25.202 GTM configuration constants**

| Module | Constant | Value |
|---|---|---|
| MCS | RAW | 9 |
| MCS | USR | 1 |
| ATOM | cCATO | 8, 4 |
| TIM | cITIM | 2 |
| TIM | cCTIM | 8,8 |

## 25.18.3 Register reset values device depended hardcoded

### 25.18.3.1 GTM0GTMREV

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00000_H$

**Value after reset:** $2072\ 12A1_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEV_CODE2 | | | | DEV_CODE1 | | | | DEV_CODE0 | | | | MAJOR | | | |
| Value after reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MINOR | | | | NO | | | | STEP | | | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **Section 25.6.9.1, GTM0GTMREV**.

### 25.18.3.2 GTM0ARUCADDREND

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $002B4_H$

**Value after reset:** $0000\ 0015_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CADDR_END | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

For detail, see **Section 25.7.6.14, GTM0ARUCADDREND**.

## 25.18.4 Register reset values defined by hardware configuration

### 25.18.4.1 GTM0GTMBRIDGEMODE

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00030_H$

**Value after reset:** $0400\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BUFF_DPT | | | | | | | | — | — | — | — | — | — | — | BRG_RST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | SYNC_INPUT_REG | — | — | BUFF_OVL | MODE_UP_PGR | — | — | — | — | — | — | MSK_WR_RSP | BRG_MODE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **Section 25.6.9.9, GTM0GTMBRIDGEMODE**.

### 25.18.4.2 GTM0GTMBRIDGEPTR1

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00034_H$

**Value after reset:** $0040\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSP_TRAN_RDY | | | | | | FBC | | | | | | ABT_TRAN_PGR | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ABT_TRAN_PGR | TRAN_IN_PGR | | | | | FIRST_RSP_PTR | | | | | NEW_TRAN_PTR | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **Section 25.6.9.10, GTM0GTMBRIDGEPTR1**.

### 25.18.4.3 GTM0GTMHWCONF

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00024_H$

**Value after reset:** $000F\ 7331_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | IRQ_MODE_SINGLE_PULSE | IRQ_MODE_PULSE_NOTIFY | IRQ_MODE_PULSE | IRQ_MODE_LEVEL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | ARU_CONNECT_CONFIG | ERM | RAM_INIT_RST | TOM_TRIG_CHAIN | | | TOM_OUT_RST | ATOM_TRIG_CHAIN | | | ATOM_OUT_RST | — | SYNC_INPUT_REG | BRIDGE_MODE_RST | GRSTEN |
| Value after reset | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **Section 25.6.9.14, GTM0GTMHWCONF**.

### 25.18.4.4 GTM0GTMATOMiOUT

**Access:** This register can be read in 32-bit units.

**Address:** GTM0GTMATOM0OUT: <GTM_base> + $00098_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ATOM_IP1_OUT_N | | | | | | | | ATOM_IP1_OUT | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ATOM_I_OUT_N | | | | | | | | ATOM_I_OUT | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **Section 25.6.9.15, GTM0GTMATOMiOUT (i = 0, 2)**.

### 25.18.4.5 GTM0ATOMixCTRL

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0ATOM00CTRL: <GTM_base> + $0D004_H$,
GTM0ATOM01CTRL: <GTM_base> + $0D084_H$,
GTM0ATOM02CTRL: <GTM_base> + $0D104_H$,
GTM0ATOM03CTRL: <GTM_base> + $0D184_H$,
GTM0ATOM04CTRL: <GTM_base> + $0D204_H$,
GTM0ATOM05CTRL: <GTM_base> + $0D284_H$,
GTM0ATOM06CTRL: <GTM_base> + $0D304_H$,
GTM0ATOM07CTRL: <GTM_base> + $0D384_H$,
GTM0ATOM20CTRL: <GTM_base> + $0E004_H$
GTM0ATOM21CTRL: <GTM_base> + $0E084_H$
GTM0ATOM22CTRL: <GTM_base> + $0E104_H$
GTM0ATOM23CTRL: <GTM_base> + $0E184_H$
GTM0ATOM24CTRL: <GTM_base> + $0E204_H$
GTM0ATOM10CTRL: <GTM_base> + $0D804_H$
GTM0ATOM11CTRL: <GTM_base> + $0D884_H$
GTM0ATOM12CTRL: <GTM_base> + $0D904_H$
GTM0ATOM13CTRL: <GTM_base> + $0D984_H$
GTM0ATOM14CTRL: <GTM_base> + $0DA04_H$
GTM0ATOM15CTRL: <GTM_base> + $0DA84_H$
GTM0ATOM16CTRL: <GTM_base> + $0DB04_H$
GTM0ATOM17CTRL: <GTM_base> + $0DB84_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | SOMB | — | — | ABM | OSM | SLA | TRIGOUT | EXTTRIGOUT | EXT_TRIG | OSM_TRIG | RST_CCU0 | — | — | — | WR_REQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CLK_SRC_SR | | | SL | — | CMP_CTRL | ACB | | | | | ARU_EN | TB12_SEL | MODE | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

For detail, see **(1) GTM0ATOMixCTRL (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)**.

### 25.18.4.6 GTM0ATOMixSTAT

**Access:** This register can be read in 32-bit units.

**Address:**
GTM0ATOM00STAT: <GTM_base> + $0D01C_H$,
GTM0ATOM01STAT: <GTM_base> + $0D09C_H$,
GTM0ATOM02STAT: <GTM_base> + $0D11C_H$,
GTM0ATOM03STAT: <GTM_base> + $0D19C_H$,
GTM0ATOM04STAT: <GTM_base> + $0D21C_H$,
GTM0ATOM05STAT: <GTM_base> + $0D29C_H$,
GTM0ATOM06STAT: <GTM_base> + $0D31C_H$,
GTM0ATOM07STAT: <GTM_base> + $0D39C_H$,
GTM0ATOM10STAT: <GTM_base> + $0D81C_H$
GTM0ATOM11STAT: <GTM_base> + $0D89C_H$
GTM0ATOM12STAT: <GTM_base> + $0D91C_H$
GTM0ATOM13STAT: <GTM_base> + $0D99C_H$
GTM0ATOM14STAT: <GTM_base> + $0DA1C_H$
GTM0ATOM15STAT: <GTM_base> + $0DA9C_H$
GTM0ATOM16STAT: <GTM_base> + $0DB1C_H$
GTM0ATOM17STAT: <GTM_base> + $0DB9C_H$
GTM0ATOM20STAT: <GTM_base> + $0E01C_H$
GTM0ATOM21STAT: <GTM_base> + $0E09C_H$
GTM0ATOM22STAT: <GTM_base> + $0E11C_H$
GTM0ATOM23STAT: <GTM_base> + $0E19C_H$
GTM0ATOM24STAT: <GTM_base> + $0E21C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ACBO | | | | | — | WRF | DV | ACBI | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **(2) GTM0ATOMixSTAT (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)**.

### 25.18.4.7 GTM0MCSiCTRLSTAT

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0MCS0CTRLSTAT: <GTM_base> + $30064_H$
GTM0MCS1CTRLSTAT: <GTM_base> + $31064_H$

**Value after reset:** $000X\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | EN_TIM_FOUT | — | — | ERR_SRC_ID | | — | — | HLT_SP_OFL | RAM_RST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SCD_CH | | | | — | — | — | — | — | — | SCD_MODE | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

For detail, see **Section 25.13.9.11, GTM0MCSiCTRLSTAT (i = 0, 1)**.

### 25.18.5 Memory Address Ranges

The following table lists the address ranges of all memory mapped RAM regions implemented in the GTM-IP. The submodule address offsets are already applied in this table. Since the GTM-IP supports a configurable RAM size for several RAM ports, the table list minimum and maximum value for the upper memory addresses, whereas the maximum addresses can be found in brackets.

**Table 25.203 Memory Address Ranges**

| Submodule | Memory Address Ranges |
|---|---|
| MCS0_MEMORY | $0003\ 8000_H$ to $0003\ 87FC_H$ |

## 25.18.6 ARU

### 25.18.6.1 ARU Write Address Overview

**Table 25.204 ARU Write Address Overview**

| Name | Address |
|---|---|
| ARU_ACCESS | $000_H$ |
| **TIM [0 .. 1]** | |
| TIM0_WRADDR[0..7] | $001_H$ to $008_H$ |
| TIM1_WRADDR[0..7] | $009_H$ to $010_H$ |
| unused | $011_H$ to $038_H$ |
| **misc** | |
| unused | $039_H$ to $076_H$ |
| **MCS [0]** | |
| MCS0_WRADDR[0..23] | $077_H$ to $08E_H$ |
| unused | $08F_H$ to $11E_H$ |
| **ATOM [0 .. 1]** | |
| ATOM0_WRADDR[0..7] | $11F_H$ to $126_H$ |
| ATOM1_WRADDR[0..3] | $127_H$ to $12A_H$ |
| unused | $12B_H$ to $17E_H$ |
| **misc** | |
| unused | $17F_H$ to $1FD_H$ |
| ARU_EMPTY_ADDR | $1FE_H$ |
| ARU_FULL_ADDR | $1FF_H$ |

### 25.18.6.2 ARU Round Trip Time

The round trip time of Device 207 is 22 * SYS_CLK for a one-ARU-slave configuration. A two-ARU-slave configuration is not implemented.

### 25.18.6.3 ARU port addressing

Each ARU connected data sink has its own specific address. The ARU caddr counter which represents the ARU roundtrip cycle is addressing each data sink. The following table describes the addressing order of the ARU-connected modules/data sinks.

**Table 25.205 ARU port addressing**

| Name | caddr |
|---|---|
| reserved | 0 |
| **ATOM-0** | |
| channel-0 | 1 |
| channel-1 | 2 |
| channel-2 | 3 |
| channel-3 | 4 |
| channel-4 | 5 |
| channel-5 | 6 |
| channel-6 | 7 |
| channel-7 | 8 |
| **MCS-0** | |
| channel-0 | 9 |
| channel-1 | 10 |
| channel-2 | 11 |
| channel-3 | 12 |
| channel-4 | 13 |
| channel-5 | 14 |
| channel-6 | 15 |
| channel-7 | 16 |
| channel-8 | 17 |
| **ATOM-1** | |
| channel-0 | 18 |
| channel-1 | 19 |
| channel-2 | 20 |
| channel-3 | 21 |
| **misc** | |
| reserved | 22..127 |

### 25.18.7 ATOM

Bit-reversed mode (PCM) in ATOM SOMP is available in the following channels:

**Table 25.206 Bit-reversed mode (PCM) in ATOM SOMP**

| Module | Ch-0 | Ch-1 | Ch-2 | Ch-3 | Ch-4 | Ch-5 | Ch-6 | Ch-7 |
|---|---|---|---|---|---|---|---|---|
| ATOM-0 | no | no | no | yes | no | no | no | yes |
| ATOM-1 | no | no | no | yes | — | — | — | — |

### 25.18.8 GTM Application constraints

If setting up an application on GTM-IP one has to take following constraints into account. Otherwise, the GTM-IP may not be able to fulfil the specified behavior.

**Table 25.207 GTM Application constraints**

| # | Module | Description | Required value | Effect, when not considered |
|---|---|---|---|---|
| 1 | MCS | Worst Case Execution Time of an N-cycle instruction. | ≤ 9*N system clock periods | MCS program execution may be too long for the application's requirements. |
| 2 | ATOM | If channel is triggered by preceding channel via TRIG_<x-1> signal, the selected CMU_CLK of both channels has to be the same | Identical CMU_CLK | The trigger of preceding channel my be lost |
| 3 | TIM | If a TIM channel x uses ARU transfer with enabled TDU, the minimal time $T_{IN}$ between two subsequent measurement cycles has to be greater than the maximum time $T_{SAMPLE}$ between two subsequent ARU read request events on TIM channel x. | $T_{IN} > T_{SAMPLE}$ (MCS: $T_{SAMPLE}$ = max. time between two (N)ARD(I) instructions for TIM channel x) | The ARU destination of TIM channel x is cannot distinguish between a measurement cycle overflow and a timeout with subsequent valid measurement cycle. |
| 4 | ARU | The ARU round trip time is device specific. | | |

# 25.19 GTM Device 208

## 25.19.1 Architecture Block Diagram

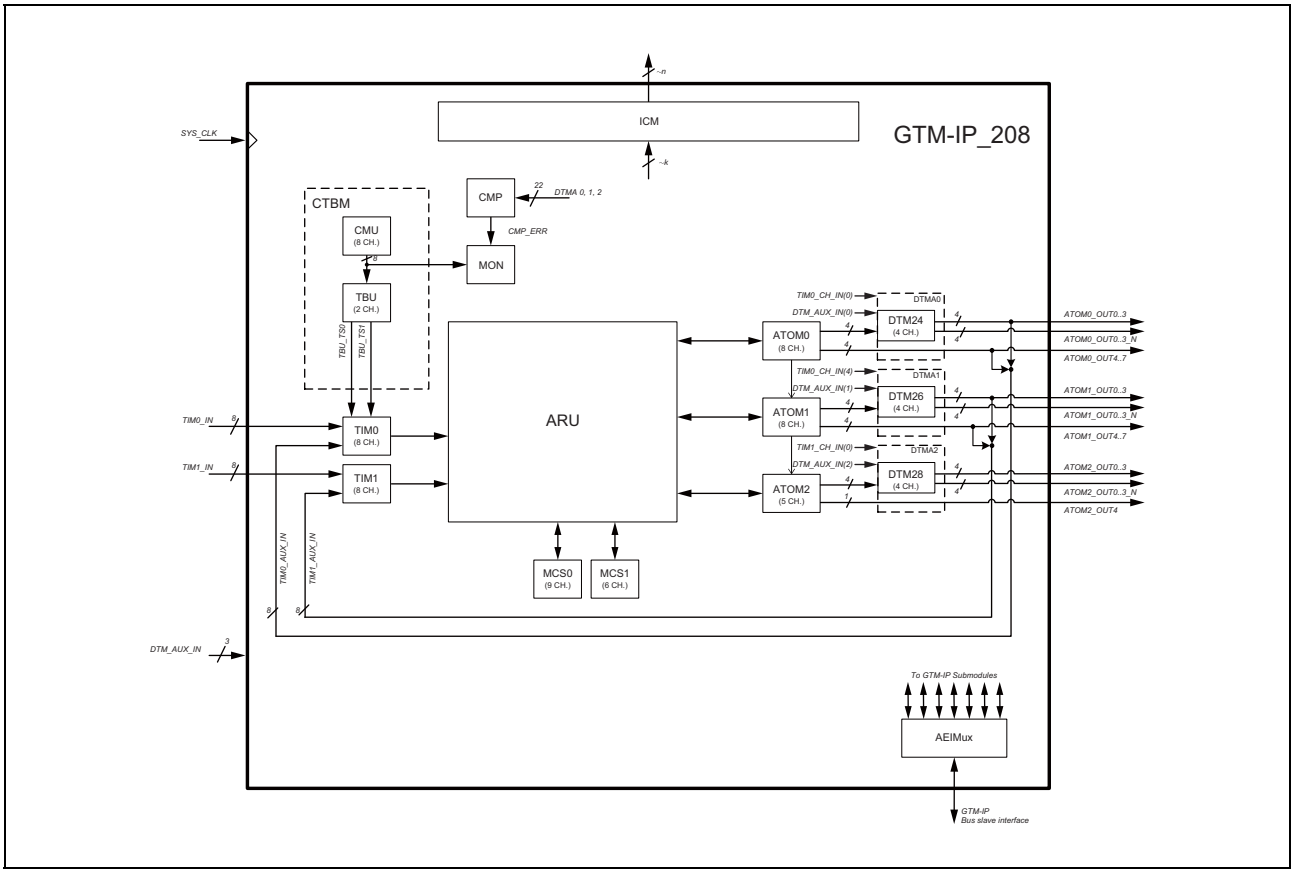



Figure 25.68 Architecture Block Diagram

### 25.19.2 GTM Device 208 Configuration

The following table lists the device configuration of device GTM-IP_208

**Table 25.208 GTM Device 208 Configuration**

| | instances | Numbering | channel |
|---|---|---|---|
| ARU | 1 | | |
| BRC | 0 | | |
| PSM | 0 | | |
| CMU | 1 | — | 8 |
| TBU | 1 | — | 2 |
| TIM | 2 | 0, 1 | 8, 8 |
| TOM | 0 | | |
| ATOM | 3 | 0, 1, 2 | 8, 8, 5 |
| DTM | 3 | 24, 26, 28 | 4, 4, 4 |
| MCS | 2 | 0, 1 | 9, 6 |
| MCFG | 1 | — | — |
| MAP | 0 | | |
| DPLL | 0 | | |
| SPE | 0 | | |
| ICM | 1 | — | — |
| CMP | 1 | — | — |
| MON | 1 | — | — |

Following configuration constants referenced in specification are defined for this device:

**Table 25.209 GTM configuration constants**

| Module | Constant | Value |
|---|---|---|
| MCS | RAW | 9 |
| MCS | USR | 1 |
| MCFG | MAW | 8 |
| MCFG | ERM | 1 |
| ATOM | cCATO | 8, 8, 5 |
| TIM | cITIM | 2 |
| TIM | cCTIM | 8,8 |

## 25.19.3 Register reset values device depended hardcoded

### 25.19.3.1 GTM0GTMREV

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00000_H$

**Value after reset:** $2082\ 12A2_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEV_CODE2 | | | | DEV_CODE1 | | | | DEV_CODE0 | | | | MAJOR | | | |
| Value after reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MINOR | | | | NO | | | | STEP | | | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **Section 25.6.9.1, GTM0GTMREV**.

### 25.19.3.2 GTM0ARUCADDREND

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $002B4_H$

**Value after reset:** $0000\ 0024_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CADDR_END | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

For detail, see **Section 25.7.6.14, GTM0ARUCADDREND**.

## 25.19.4 Register reset values defined by hardware configuration

### 25.19.4.1 GTM0GTMBRIDGEMODE

**Access:** This register can be read/written in 32-bit units.

**Address:** <GTM_base> + $00030_H$

**Value after reset:** $0400\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BUFF_DPT | | | | | | | | — | — | — | — | — | — | — | BRG_RST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | SYNC_INPUT_REG | — | — | BUFF_OVL | MODE_UP_PGR | — | — | — | — | — | — | MSK_WR_RSP | BRG_MODE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **Section 25.6.9.9, GTM0GTMBRIDGEMODE**.

### 25.19.4.2 GTM0GTMBRIDGEPTR1

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00034_H$

**Value after reset:** $0040\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSP_TRAN_RDY | | | | | | FBC | | | | | | ABT_TRAN_PGR | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ABT_TRAN_PGR | TRAN_IN_PGR | | | | | FIRST_RSP_PTR | | | | | NEW_TRAN_PTR | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **Section 25.6.9.10, GTM0GTMBRIDGEPTR1**.

### 25.19.4.3 GTM0GTMHWCONF

**Access:** This register can be read in 32-bit units.

**Address:** <GTM_base> + $00024_H$

**Value after reset:** $000F\ 7331_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | IRQ_MODE_SINGLE_PULSE | IRQ_MODE_PULSE_NOTIFY | IRQ_MODE_PULSE | IRQ_MODE_LEVEL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | ARU_CONNECT_CONFIG | ERM | RAM_INIT_RST | TOM_TRIG_CHAIN | | | TOM_OUT_RST | ATOM_TRIG_CHAIN | | | ATOM_OUT_RST | — | SYNC_INPUT_REG | BRIDGE_MODE_RST | GRSTEN |
| Value after reset | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **Section 25.6.9.14, GTM0GTMHWCONF**.

### 25.19.4.4 GTM0GTMATOMiOUT

**Access:** This register can be read in 32-bit units.

**Address:** GTM0GTMATOM0OUT: <GTM_base> + $00098_H$
GTM0GTMATOM2OUT: <GTM_base> + $0009C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ATOM_IP1_OUT_N | | | | | | | | ATOM_IP1_OUT | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ATOM_I_OUT_N | | | | | | | | ATOM_I_OUT | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **Section 25.6.9.15, GTM0GTMATOMiOUT (i = 0, 2)**.

### 25.19.4.5 GTM0ATOMixCTRL

**Access:** This register can be read/written in 32-bit units.

**Address:**
GTM0ATOM00CTRL: <GTM_base> + $0D004_H$,
GTM0ATOM01CTRL: <GTM_base> + $0D084_H$,
GTM0ATOM02CTRL: <GTM_base> + $0D104_H$,
GTM0ATOM03CTRL: <GTM_base> + $0D184_H$,
GTM0ATOM04CTRL: <GTM_base> + $0D204_H$,
GTM0ATOM05CTRL: <GTM_base> + $0D284_H$,
GTM0ATOM06CTRL: <GTM_base> + $0D304_H$,
GTM0ATOM07CTRL: <GTM_base> + $0D384_H$,
GTM0ATOM10CTRL: <GTM_base> + $0D804_H$
GTM0ATOM11CTRL: <GTM_base> + $0D884_H$
GTM0ATOM12CTRL: <GTM_base> + $0D904_H$
GTM0ATOM13CTRL: <GTM_base> + $0D984_H$
GTM0ATOM14CTRL: <GTM_base> + $0DA04_H$
GTM0ATOM15CTRL: <GTM_base> + $0DA84_H$
GTM0ATOM16CTRL: <GTM_base> + $0DB04_H$
GTM0ATOM17CTRL: <GTM_base> + $0DB84_H$
GTM0ATOM20CTRL: <GTM_base> + $0E004_H$
GTM0ATOM21CTRL: <GTM_base> + $0E084_H$
GTM0ATOM22CTRL: <GTM_base> + $0E104_H$
GTM0ATOM23CTRL: <GTM_base> + $0E184_H$
GTM0ATOM24CTRL: <GTM_base> + $0E204_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | SOMB | — | — | ABM | OSM | SLA | TRIGOUT | EXTTRIGOUT | EXT_TRIG | OSM_TRIG | RST_CCU0 | — | — | — | WR_REQ |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CLK_SRC_SR | | | SL | — | CMP_CTRL | ACB | | | | | ARU_EN | TB12_SEL | MODE | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

For detail, see **(1) GTM0ATOMixCTRL (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)**.

### 25.19.4.6 GTM0ATOMixSTAT

**Access:** This register can be read in 32-bit units.

**Address:**
GTM0ATOM00STAT: <GTM_base> + $0D01C_H$,
GTM0ATOM01STAT: <GTM_base> + $0D09C_H$,
GTM0ATOM02STAT: <GTM_base> + $0D11C_H$,
GTM0ATOM03STAT: <GTM_base> + $0D19C_H$,
GTM0ATOM04STAT: <GTM_base> + $0D21C_H$,
GTM0ATOM05STAT: <GTM_base> + $0D29C_H$,
GTM0ATOM06STAT: <GTM_base> + $0D31C_H$,
GTM0ATOM07STAT: <GTM_base> + $0D39C_H$,
GTM0ATOM10STAT: <GTM_base> + $0D81C_H$
GTM0ATOM11STAT: <GTM_base> + $0D89C_H$
GTM0ATOM12STAT: <GTM_base> + $0D91C_H$
GTM0ATOM13STAT: <GTM_base> + $0D99C_H$
GTM0ATOM14STAT: <GTM_base> + $0DA1C_H$
GTM0ATOM15STAT: <GTM_base> + $0DA9C_H$
GTM0ATOM16STAT: <GTM_base> + $0DB1C_H$
GTM0ATOM17STAT: <GTM_base> + $0DB9C_H$
GTM0ATOM20STAT: <GTM_base> + $0E01C_H$
GTM0ATOM21STAT: <GTM_base> + $0E09C_H$
GTM0ATOM22STAT: <GTM_base> + $0E11C_H$
GTM0ATOM23STAT: <GTM_base> + $0E19C_H$
GTM0ATOM24STAT: <GTM_base> + $0E21C_H$

**Value after reset:** $0000\ 000x_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ACBO | | | | | — | WRF | DV | ACBI | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

For detail, see **(2) GTM0ATOMixSTAT (i = 0, x = 0 to 7, i = 1, x = 0 to 7, i = 2, x = 0 to 4)**.

### 25.19.4.7 GTM0MCSiCTRLSTAT

**Access:** This register can be read/written in 32-bit units.

**Address:** GTM0MCS0CTRLSTAT: <GTM_base> + $30064_H$
GTM0MCS1CTRLSTAT: <GTM_base> + $31064_H$

**Value after reset:** $000x\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | EN_TIM_FOUT | — | — | ERR_SRC_ID | | — | — | HLT_SP_OFL | RAM_RST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SCD_CH | | | | — | — | — | — | — | — | SCD_MODE | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

For detail, see **Section 25.13.9.11, GTM0MCSiCTRLSTAT (i = 0, 1)**.

### 25.19.5 Memory Address Ranges

The following table lists the address ranges of all MCS memory mapped RAM regions implemented in the GTM-IP. The submodule address offsets are already applied in this table. Since the GTM-IP supports a configurable RAM size for several RAM ports, the table list minimum and maximum value for the upper memory addresses, whereas the maximum addresses can be found in brackets.

**Table 25.210 Memory Address Ranges**

| Submodule | Memory Address Ranges |
|---|---|
| MCS0_MEMORY | 0003 $8000_H$ to 0003 $87FC_H$ (0003 $8BFC_H$) |
| MCS1_MEMORY | 0004 $0000_H$ to 0004 $07FC_H$ (0004 $0BFC_H$) |

## 25.19.6 ARU

### 25.19.6.1 ARU Write Address Overview

Table 25.211 ARU Write Address Overview

| Name | Address |
|---|---|
| ARU_ACCESS | $000_H$ |
| **TIM [0 .. 1]** | |
| TIM0_WRADDR[0..7] | $001_H$ to $008_H$ |
| TIM1_WRADDR[0..7] | $009_H$ to $010_H$ |
| unused | $011_H$ to $038_H$ |
| **misc** | |
| unused | $039_H$ to $076_H$ |
| **MCS [0 .. 1]** | |
| MCS0_WRADDR[0..23] | $077_H$ to $08E_H$ |
| MCS1_WRADDR[0..23] | $08F_H$ to $0A6_H$ |
| unused | $0A7_H$ to $11E_H$ |
| **ATOM [0 .. 2]** | |
| ATOM0_WRADDR[0..7] | $11F_H$ to $126_H$ |
| ATOM1_WRADDR[0..7] | $127_H$ to $12E_H$ |
| ATOM2_WRADDR[0..4] | $12F_H$ to $133_H$ |
| unused | $0x134_H$ to $17E_H$ |
| **misc** | |
| unused | $0x17F_H$ to $1FD_H$ |
| ARU_EMPTY_ADDR | $0x1FE_H$ |
| ARU_FULL_ADDR | $0x1FF_H$ |

### 25.19.6.2 ARU Round Trip Time

The round trip time of Device 208 is 37 * SYS_CLK for a one-ARU-slave configuration and 20 * SYS_CLK for a two-ARU-slave configuration.

### 25.19.6.3 ARU port partitioning

Depending on the ARU connectivity configuration in the gtm_config_pack.vhd (aru_connect_config_c) the submodules ATOM0..2 and MCS0..1 are connected to different ARU ports as described in the table below.

The real configuration can be determined by evaluating bit ARU_CONNECT_CONFIG of register GTM_HW_CONF.

**Table 25.212 ARU port partitioning**

| | aru_connect_config_c = 0 | | aru_connect_config_c = 1 | |
|---|---|---|---|---|
| | port 1 | port 2 | port 1 | (port 2) |
| ATOM-0 | x | | x | |
| ATOM-1 | | x | x | |
| ATOM-2 | | x | x | |
| MCS-0 | x | | x | |
| MCS-1 | | x | x | |

### 25.19.6.4 ARU port addressing

Each ARU connected data sink has its own specific address. The ARU caddr counter which represents the ARU roundtrip cycle is addressing each data sink. The following table describes the addressing order of the ARU-connected modules/data sinks.

**Table 25.213 ARU port addressing (1/2)**

| Name | caddr |
|---|---|
| reserved | 0 |
| **ATOM-0** | |
| channel-0 | 1 |
| channel-1 | 2 |
| channel-2 | 3 |
| channel-3 | 4 |
| channel-4 | 5 |
| channel-5 | 6 |
| channel-6 | 7 |
| channel-7 | 8 |
| **MCS-0** | |
| channel-0 | 9 |
| channel-1 | 10 |
| channel-2 | 11 |
| channel-3 | 12 |
| channel-4 | 13 |
| channel-5 | 14 |
| channel-6 | 15 |
| channel-7 | 16 |
| channel-8 | 17 |

Table 25.213 ARU port addressing (2/2)

| Name | caddr |
|---|---|
| **ATOM-1** | |
| channel-0 | 18 |
| channel-1 | 19 |
| channel-2 | 20 |
| channel-3 | 21 |
| channel-4 | 22 |
| channel-5 | 23 |
| channel-6 | 24 |
| channel-7 | 25 |
| **ATOM-2** | |
| channel-0 | 26 |
| channel-1 | 27 |
| channel-2 | 28 |
| channel-3 | 29 |
| channel-4 | 30 |
| **MCS-1** | |
| channel-0 | 31 |
| channel-1 | 32 |
| channel-2 | 33 |
| channel-3 | 34 |
| channel-4 | 35 |
| channel-5 | 36 |

### 25.19.7 ATOM

Bit-reversed mode (PCM) in ATOM SOMP is available in the following channels:

**Table 25.214 Bit-reversed mode (PCM) in ATOM SOMP**

| Module | Ch-0 | Ch-1 | Ch-2 | Ch-3 | Ch-4 | Ch-5 | Ch-6 | Ch-7 |
|---|---|---|---|---|---|---|---|---|
| ATOM-0 | no | no | no | yes | no | no | no | yes |
| ATOM-1 | no | no | no | yes | no | no | no | yes |
| ATOM-2 | no | no | no | yes | no | — | — | — |

### 25.19.8 GTM Application constraints

If setting up an application on GTM-IP one has to take following constraints into account. Otherwise, the GTM-IP may not be able to fulfil the specified behavior.

**Table 25.215 GTM Application constraints**

| # | Module | Description | Required value | Effect, when not considered |
|---|---|---|---|---|
| 1 | MCS | Worst Case Execution Time of an N-cycle instruction. | <= 9*N system clock periods | MCS program execution may be too long for the application's requirements. |
| 2 | ATOM | If channel is triggered by preceding channel via TRIG_<x-1> signal, the selected CMU_CLK of both channels has to be the same | Identical CMU_CLK | The trigger of preceding channel my be lost |
| 3 | TIM | If a TIM channel x uses ARU transfer with enabled TDU, the minimal time $T_{IN}$ between two subsequent measurement cycles has to be greater than the maximum time $T_{SAMPLE}$ between two subsequent ARU read request events on TIM channel x. | $T_{IN} > T_{SAMPLE}$ (MCS: $T_{SAMPLE}$ = max. time between two (N)ARD(I) instructions for TIM channel x) | The ARU destination of TIM channel x is cannot distinguish between a measurement cycle overflow and a timeout with subsequent valid measurement cycle. |
| 4 | ARU | The ARU round trip time is device specific. | | |

## 25.20 CAUTION

| No. | Outline |
|---|---|
| ID1 | GTM - GTM_HALT has static 5 clocks latency<br>(Direction of use) |
| ID2 | GTM - TOM/ATOM SOMP mode: initial delay of one shot pulse triggered by preceding channel<br>(Direction of use) |
| ID3 | GTM - MCS: Evaluation of CAT bit after blocking ARU instruction<br>(Direction of use) |
| ID4 | GTM - TIM: Incorrect signal level bit ECNT[0] in mode TIEM, TPWM, TIPM, TPIM, TGPS<br>(Direction of use) |
| ID5 | GTM - DTM: cross channel dead time does not prevent two cross linked outputs to be high<br>(Direction of use) |
| ID6 | GTM - (A)TOM: no CCU1 interrupt in case of CM1 = 0 or 1 and RST_CCU0 = 1<br>(Direction of use) |
| ID7 | GTM - TIM: incorrect signal level on TIM_MODE change if TIM channel is disabled<br>(Direction of use) |
| ID8 | GTM - TIM: unexpected CNTS register update in TPWM OSM mode<br>(Direction of use) |
| ID9 | GTM - TOM/ATOM: no update of CM0/CM1/CLK_SRC via trigger signal from preceding instance if selected CMU_CLKx is not SYS_CLK<br>(Direction of use) |
| ID10 | GTM - ATOM: data loss in SOMS one-shot mode if ARU is enabled and the period of the selected CMU_CLKx is greater than ARU-cycle-time/2.<br>(Direction of use) |
| ID11 | GTM - DTM: shut-off priority causes spikes at DTM output<br>(Direction of use) |
| ID12 | GTM - ATOM SOMB mode: if ARU_EN = 1, a force update does not update ACBI flags but uses value from ATOM_CH_CTRL register<br>(Technical limitation) |
| ID13 | GTM - TOM/ATOM: async. update in SOMP mode with CM1 = 0 and selected CMU clock unequal sys_clk not functional<br>(Direction of use) |
| ID14 | GTM - (A)TOM: output signal is postponed one period for the values CM0 = 1 and CM1 > CM0 if CN0 is reset by the trigger of a preceding channel (RST_CCU0 = 1)<br>(Direction of use) |

| | | |
|---|---|---|
| ID1 | | GTM - GTM_HALT has static 5 clocks latency |
| | Description | When GTM_HALT related event is detected by GTM TEU module, the GTM system stops after 5 clocks. |
| | Workaround | None |
| ID2 | | GTM - TOM/ATOM SOMP mode: initial delay of one shot pulse triggered by preceding channel |
| | Description | In case of following configuration of TOM/AOM SOMP mode, the first pulse on trigger from preceding channel is not generated:<br>The triggered channel is configured to<br>• RST_CCU0 = 0,<br>• OSM = 1,<br>• OSM_TRIG = 1,<br>• CN0 = MAX-1,<br>And the preceding channel which triggers the single pulse generation is configured to<br>• TRIGOUT = 1<br>• CM0 = MAX -> the triggering channel counts from 0 to MAX-1 until it triggers<br><br>(Description is based of Errata description GTM-IP-176) |
| | Workaround | Preset CN0 of triggered channel to CN0 < MAX-1 |
| ID3 | | GTM - MCS: Evaluation of CAT bit after blocking ARU instruction |
| | Description | The specification for the instructions ARD, AWR, ARDI, and AWRI claims that the CAT bit can be evaluated by the MCS program in order to check if the last ARU transfer was successful (CAT=0) or cancelled by Software (CAT=1). However, since the CAT bit can be set directly by Software to cancel an ARU transfer at any time the bit does not reflect the status information reliably. Bad case: If the CPU software is setting CAT between the time of ARU data arrival and evaluation of CAT bit.<br>(Description is based of Errata description GTM-IP-178) |
| | Workaround | If the mechanism for cancelling blocking ARU transfers by CPU is used and data consistency by ARU transfers is important, a possible workaround may check the consistency by inspection of the transferred data (e.g. checking for linear increment of ECNT for data transfers from TIM to MCS). |
| ID4 | | GTM - TIM: Incorrect signal level bit ECNT[0] in mode TIEM, TPWM, TIPM, TPIM, TGPS |
| | Description | In case of re-enabling a previously disabled TIM channel the bit ECNT[0] might not reflect the actual signal level of the corresponding input TIM[i]_CH[x]_FOUT until the next input edge occurs. This situation can only occur if between disabling and re-enabling the ECNT register is not read.<br>(Description is based of Errata description GTM-IP-181) |
| | Workaround | 1. After disabling the TIM channel, ensure that the ECNT register is read at least once and afterwards the TIM channel can be re-enabled<br>2. Before re-enabling a TIM channel, issue a TIM channel reset to reconfigure the TIM channel control registers. |

| | | |
|---|---|---|
| ID5 | | GTM - DTM: cross channel dead time does not prevent two cross linked outputs to be high |
| | Description | In case of full/half bridge controlling where the low side switch signal is not the inverse signal of the high side switch signal, one can use the DTM cross channel dead time mode to introduce dead time. Then two neighbored TOM/ATOMchannel generate two PWM signals at the DTM inputs DTM_IN[2x] and DTM_IN[2x+1] and the DTM should introduce the dead time wherever necessary and avoid enabling of both output signals at the same point in time. The final output signals are available at DTM_OUT[2x] and DTM_OUT[2x+1]. If both DTM inputs have their edges at the same point in time (typically, one has arising edge while the other one has a falling edge) the correct dead time is applied. If it happens that both DTM input signals are high over a common time period, the DTM erroneously does not prevent from setting both outputs DTM_OUT[2x] and DTM_OUT[2x+1].<br>In typical applications this causes a damage of the switches.<br>(Description is based of Errata description GTM-IP-201) |
| | Workaround | In case of DTM cross channel mode enabled, avoid by TOM/ATOM signal generation that both cross linked DTM inputs DTM_IN[2x] and DTM_IN[2x+1] are high (i.e. '1') at the same point in time. |
| ID6 | | GTM - (A)TOM: no CCU1 interrupt in case of CM1 = 0 or 1 and RST_CCU0 = 1 |
| | Description | In case of channel x has configuration of RST_CCU0=1 (i.e. CN0 is reset by trigger input) and CN0 counts from 0 to MAX:<br>• if CM1 = 0, CM0>0 -> no CCU1 interrupt is generated<br>• if CM1 = 1, CM0=MAX+1 -> only one time a CCU1 interrupt is generated<br>(Description is based of Errata description GTM-IP-202) |
| | Workaround | Use for triggering channel y (i.e. the channel that triggers on channel x the reset of counter CN0) the configuration of CM0 = MAX, CM1 = 1.<br>In case of duty cycle configuration of CM1=0 and CM0>0 on channel x use instead of CCU1 interrupt on channel x the CCU0 interrupt of triggering channel y.<br>In case of duty cycle configuration of CM1 = 1 and CM0 = MAX + 1 on channel x use instead of CCU1 interrupt on channel x the CCU1 interrupt of triggering channel y. |
| ID7 | | GTM - TIM: incorrect signal level on TIM_MODE change if TIM channel is disabled |
| | Description | If TIM_EN = 0 and TIM_MODE = "100" (TBCM) and corresponding channel input signal is high any write of TIM_MODE! = "100" while TIM_EN = 0 will not update the signal level bit ECNT[0]. Expected operation is that ECNT[0] will be set to the actual channel input value on TIM_MODE change.<br>(Description is based of Errata description GTM-IP-204) |
| | Workaround | Never set unnecessary TIM_MODE = "100" followed by TIM_MODE! = "100" while TIM_EN = 0. |
| ID8 | | GTM - TIM: unexpected CNTS register update in TPWM OSM mode |
| | Description | If OSM = 1 and TIM_MODE = "000" (TPWM) an active edge defined by DSL will stop the measurement. In case of an inactive edge following after 1 GTM system clock cycle the active edge the CNTS register will be reset unexpected.<br>(Description is based of Errata description GTM-IP-205) |
| | Workaround | 1. Use CMU clock in TIM channel with frequency lesser than system clock<br>2. Enable filter and configure filter parameter in a way that two consecutive edges will never occur with distance of GTM system clock. |
| ID9 | | GTM - TOM/ATOM: no update of CM0/CM1/CLK_SRC via trigger signal from preceding instance if selected CMU_CLKx is not SYS_CLK |
| | Description | No update of CM0, CM1 and CLK_SRC is done although the update is enabled by register TOM[i]_TGC[y]_GLB_CTRL / ATOM[i]_AGC_GLB_CTRL if selected CMU_CLKx is not SYS_CLK.<br>(Description is based of Errata description GTM-IP-209) |
| | Workaround | One of the following workaround should be applied:<br>1. The channel of instance i+1 that should be triggered has to use a clock of period identical to SYS_CLK period<br>2. Set up on instance i+1 a redundant channel to trigger other channel of instance i+1 like it was set up on instance i to trigger other channel. Then, start both instances synchronously by using the TBU time base comparator of AGC/TGCx unit (i.e. the ATOM[i]_AGC_ATC_TB / TOM[i]_TGC[y]_ACT_TB register) |

| | | |
|---|---|---|
| ID10 | | GTM - ATOM: data loss in SOMS one-shot mode if ARU is enabled and the period of the selected CMU_CLKx is greater than ARU-cycle-time/2. |
| | Description | ATOM in SOMS one-shot mode starts to requests new data from ARU with ARU_EN = 1. If new data is delivered by ARU and stored into SR0/1 register, the data will be transferred to CM0/1 register and the ATOM starts to shift with next selected CMU_CLKx. In parallel ATOM requests immediately new data from ARU.<br>If ARU will deliver next data before the first bit of the first data is shifted out which means before the next CMU_CLKx takes place, the data will be stored into SR0/1 register but it will not be marked as valid (bit DV not set) and therefore it will be ignored.<br>(Description is based of Errata description GTM-IP-210) |
| | Workaround | It has to be ensured, that the time between delivering of two new data from ARU is greater than CMU_CLKx periods. This can be reached by delivering the data by MCS instead of by FIFO.<br>The issue can only occur if the ARU roundtrip time is greater than 2 CMU_CLKx periods. |
| ID11 | | GTM - DTM: shut-off priority causes spikes at DTM output |
| | Description | If DTM shut-off feature is configured with GTM0DTMiCTRL.UPD_MODE = 0b010 and GTM0DTMiCTRL.DTM_SEL = 0b00 .. 0b11 an edge at input DTM[i-1]_IN0_REDGE, DTM[i-1]_IN0_FEDGE, IN0_REDGE or IN0_FEDGE while shutoff is active (i.e. signal shut_off is high) cause a small pulse at DTM output DTM_OUT[x]_N.<br>Which edge type on input DTM[i-1]_IN0_REDGE, DTM[i-1]_IN0_FEDGE, IN0_REDGE or IN0_FEDGE triggers a pulse at the output depends on DTM_SEL.<br>If DTM shut-off feature is configured with UPD_MODE = 0b001 an edge at TIM_CH_IN0, TIM_CH_IN1 or DTM_AUX_IN or DTM[i-1]_PSU_IN while shut-off is active (i.e. signal shut_off is high) cause a small pulse at DTM output DTM_OUT[x]_N.<br>The root cause of the problem is the specified behavior for DTM:<br>"Note: The reset of SHUT_OFF_SYNC has higher priority than the set of this signal." |
| | Workaround | Do not use GTM0DTMiCTRL.UPD_MODE = 0b010 or 0b001 for reset of shut-off. |
| ID12 | | GTM - ATOM SOMB mode: if ARU_EN=1, a force update does not update ACBI flags but uses value from ATOM_CH_CTRL register |
| | Description | In case of ATOM SOMB mode and ARU_EN=1, if the channel has received new values via ARU and the new values were stored in register SR0, SR1 and ACB_SR register, a force update does not update the ACBI register with the content of register ACB_SR. Instead, the value of register ATOM_CH_CTRL is updated to ACBI register. |
| | Workaround | Do not used a forced update if data for ATOM is provided via ARU (ARU_EN = 1). |
| ID13 | | GTM - TOM/ATOM: async. update in SOMP mode with CM1 = 0 and selected CMU clock unequal sys_clk not functional |
| | Description | An asynchronous update of the duty cycle by writing value 0 to CM1 register while a CMU clock unequal sys_clk is selected is not working. It is expected that the output signal level is set immediately to inactive level but it will remain at actual level. |
| | Workaround | Writing value 1 instead of 0 to CM1 register. |
| ID14 | | GTM - (A)TOM: output signal is postponed one period for the values CM0 = 1 and CM1 > CM0 if CN0 is reset by the trigger of a preceding channel (RST_CCU0 = 1) |
| | Description | If counter CN0 is reset by the trigger of a preceding channel (bit RST_CCU0 of register TOM[i]_CH[x]_CLRL/ATOM[i]_CH[x]_CTRL is set), then the value of CM0 defines the signal edge to SL (signal level), whereas CM1 defines the edge to !SL (inverted signal level).<br>If – in this case – the value 1 is configured for the output edge to SL (CM0 = 1) and CM1 is configured to greater than CM0 (CM1 > CM0), the expected output edge will be postponed by one period. |
| | Workaround | Instead of configuring CM0 = 1, it is also possible to configure CM1 = 1 and to invert SL to get the expected edge at counter value 1 (CN0 = 1). |

## 25.21 Difference among P1M-C, P1H-C and P1H-CE

There are differences in channels, interrupts, and external pins. For details, See **Table 25.2**, **Table 25.5**, and **Table 25.6**.

# Section 26 Peripheral Interconnect (PIC)

## 26.1 Feature

This section contains a generic description of the Peripheral Interconnection (PIC). The first part of this section describes all RH850/P1x-C specific properties, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of the PIC (PIC2C).

### 26.1.1 Number of Units and Channels

This device has the following number of units of the PIC.

**Table 26.1 Units and Channels**

| Product | P1M-C, P1H-C, P1H-CE |
|---|---|
| Number of Units | 1 |
| Name | PIC2C |

### 26.1.2 Register Base Address

PIC base addresses are listed in the following table. PIC register addresses are given as offsets from the base addresses in general.

**Table 26.2 Register Base Address**

| Base Address Name | Base Address |
|---|---|
| <PIC2C_base> | FFD6 $8000_H$ |

### 26.1.3 Clock Supply

Supply clocks by and to PIC are listed in the following table.

**Table 26.3 Clock Supply**

| Explanation | Specification |
|---|---|
| All of module operations | High speed system clock: CLK_HSB |

### 26.1.4 Interrupt and DMA/DTS Requests

This module has no interrupt and DMA/DTS requests.

### 26.1.5 External Input and Output Pins

The table below shows the pin function information.

**Table 26.4 Pin Function Information**

| Pin Name | I/O | Description | Supporting Device | |
|---|---|---|---|---|
| | | | P1M-C, P1H-C (4MB, BGA-156) | P1H-C (4MB, BGA-292), P1H-C (8MB), P1H-CE |
| ESO0Z | I | Hi-Z control for GTM output | √ | √ |
| ESO1Z | | | √ | √ |
| ESO2Z | | | — | √ |

# 26.2 Overview

## 26.2.1 Functional Overview

PIC connects some peripherals with each other in order to achieve enhanced functionality of a stand-alone function.

PIC has following features:

- ADCF trigger select function
- Signal routing function for GTM:
  - ADCF conversion interrupt routed to GTM input
  - Baud rate measurement for an UART (RLIN3)
  - Hi-Z control function over external pin for GTM output
  - GTM output monitor for PWM diagnostic

## 26.2.2 Block Diagram

The following figure shows the block diagram.

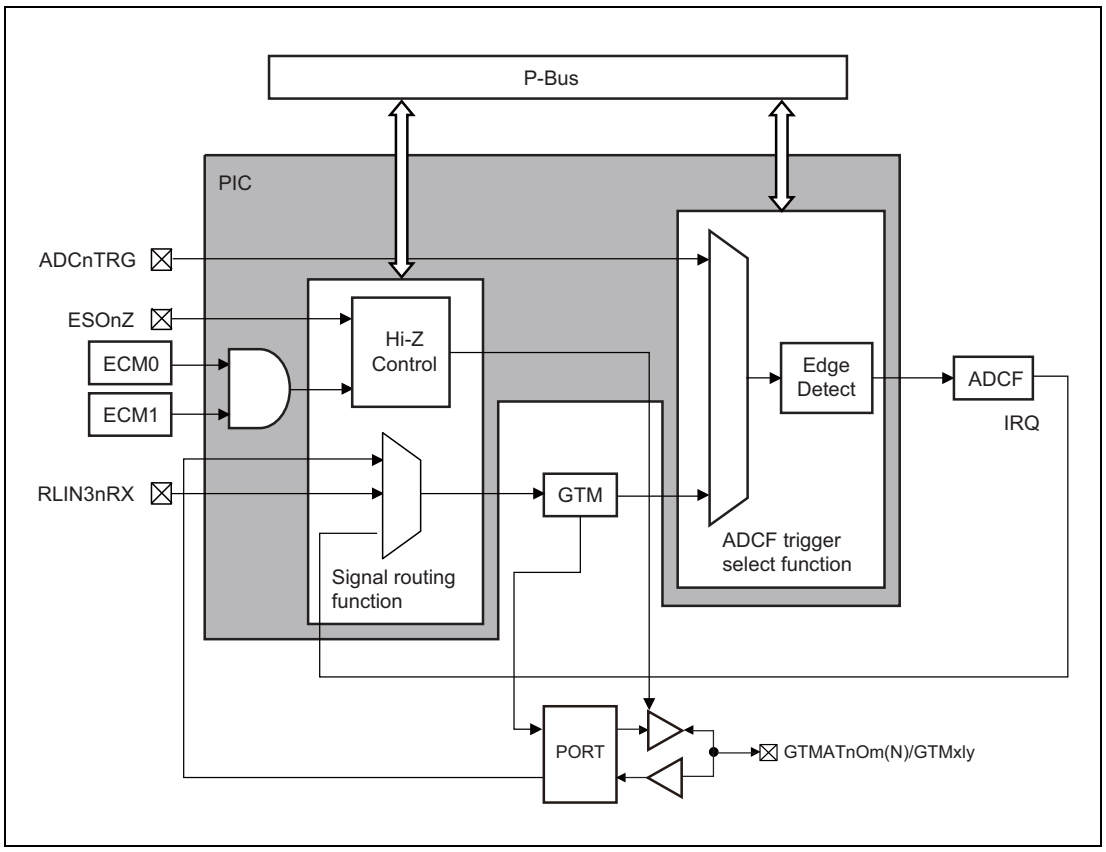



**Figure 26.1 Block diagram of PIC**

# 26.3 Registers

## 26.3.1 List of Registers

Table 26.5 List of Registers

| Address Offset*1 | Register Name | Description | Access Width | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | Other |
| $00_H$ | PIC2CTRGPREMUX0 | A/D converter trigger select 0 | 32 | 0000 $0000_H$ | PBG3#0.PG3-one | — |
| $04_H$ | PIC2CTRGPREMUX1 | A/D converter trigger select 1 | 32 | 0000 $0000_H$ | PBG3#0.PG3-one | — |
| $10_H$ | PIC2CTRGMUX0 | A/D converter trigger select control 0 | 32 | 0000 $0000_H$ | PBG3#0.PG3-one | — |
| $14_H$ | PIC2CTRGMUX1 | A/D converter trigger select control 1 | 32 | 0000 $0000_H$ | PBG3#0.PG3-one | — |
| $18_H$ | PIC2CEDGSEL0 | A/D converter trigger edge control 0 | 32 | 0000 $0000_H$ | PBG3#0.PG3-one | — |
| $1C_H$ | PIC2CEDGSEL1 | A/D converter trigger edge control 1 | 32 | 0000 $0000_H$ | PBG3#0.PG3-one | — |
| $20_H$ | PIC2CENP2TIM0 | Path to TIM0 enable control register | 32 | 0000 $0000_H$ | PBG3#0.PG3-one | — |
| $24_H$ | PIC2CENP2TIM1 | Path to TIM1 enable control register | 32 | 0000 $0000_H$ | PBG3#0.PG3-one | — |
| $28_H$ | PIC2CENHIZDTM | Hi-Z function for GTM output enable control register | 32 | 0000 $0000_H$ | PBG3#0.PG3-one | — |

Note 1. The base address is described in **Table 26.2**.

**NOTE**

An unintended pulse may be generated and propagated into connected modules when switching any PIC register settings. So each PIC register setting should be changed only during the relevant function of connected module disabled.

Table 26.6 Register Reset Condition

| Register Name | Reset Condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| All registers | √ | √ | √ | √ | — |

## 26.3.2 PIC2CTRGPREMUXn — A/D converter trigger select n (n = 0/1)

The PIC2CTRGPREMUX0 and PIC2CTRGPREMUX1 registers select GTM triggers respectively for ADCF0 and ADCF1.

**Access:** This register can be read/written in 32-bit units

**Address:** PIC2CTRGPREMUX0 <PIC2C_base> + $00_H$
PIC2CTRGPREMUX1 <PIC2C_base> + $04_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | PREMUXn4 | | | | | | PREMUXn3 | | | | | | PREMUXn2 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PREMUXn2 | | | | PREMUXn1 | | | | | | PREMUXn0 | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 26.7 PIC2CTRGPREMUXn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | When read, the value after reset is read. Writing is ignored. |
| 29 to 24 | PREMUXn4 | Selects GTM trigger for ADCFn scan group 4 |
| 23 to 18 | PREMUXn3 | Selects GTM trigger for ADCFn scan group 3 |
| 17 to 12 | PREMUXn2 | Selects GTM trigger for ADCFn scan group 2 |
| 11 to 6 | PREMUXn1 | Selects GTM trigger for ADCFn scan group 1 |
| 5 to 0 | PREMUXn0 | Selects GTM trigger for ADCFn scan group 0 |

Regarding the selected trigger, see the table below.

**Table 26.8 GTM trigger source selection by PREMUXnx bits; x means scan group: 0 to 4 (1/3)**

| PREMUXnx | Which GTM trigger source selected for ADCF SGx | Supporting device | |
|---|---|---|---|
| | | P1M-C | P1H-C, P1H-CE |
| $00_H$ | TIM0_IRQ_0 | √ | √ |
| $01_H$ | TIM0_IRQ_1 | √ | √ |
| $02_H$ | TIM0_IRQ_2 | √ | √ |
| $03_H$ | TIM0_IRQ_3 | √ | √ |
| $04_H$ | TIM0_IRQ_4 | √ | √ |
| $05_H$ | TIM0_IRQ_5 | √ | √ |
| $06_H$ | TIM0_IRQ_6 | √ | √ |
| $07_H$ | TIM0_IRQ_7 | √ | √ |
| $08_H$ | TIM1_IRQ_0 | √ | √ |
| $09_H$ | TIM1_IRQ_1 | √ | √ |
| $0A_H$ | TIM1_IRQ_2 | √ | √ |
| $0B_H$ | TIM1_IRQ_3 | √ | √ |
| $0C_H$ | TIM1_IRQ_4 | √ | √ |

Table 26.8 GTM trigger source selection by PREMUXnx bits; x means scan group: 0 to 4 (2/3)

| PREMUXnx | Which GTM trigger source selected for ADCF SGx | Supporting device | |
|---|---|---|---|
| | | P1M-C | P1H-C, P1H-CE |
| $0D_H$ | TIM1_IRQ_5 | √ | √ |
| $0E_H$ | TIM1_IRQ_6 | √ | √ |
| $0F_H$ | TIM1_IRQ_7 | √ | √ |
| $10_H$ | ATOM0_OUT_0 | √ | √ |
| $11_H$ | ATOM0_OUT_0_N | √ | √ |
| $12_H$ | ATOM0_OUT_1 | √ | √ |
| $13_H$ | ATOM0_OUT_1_N | √ | √ |
| $14_H$ | ATOM0_OUT_2 | √ | √ |
| $15_H$ | ATOM0_OUT_2_N | √ | √ |
| $16_H$ | ATOM0_OUT_3 | √ | √ |
| $17_H$ | ATOM0_OUT_3_N | √ | √ |
| $18_H$ | ATOM0_OUT_4 | √ | √ |
| $19_H$ | ATOM0_OUT_5 | √ | √ |
| $1A_H$ | ATOM0_OUT_6 | √ | √ |
| $1B_H$ | ATOM0_OUT_7 | √ | √ |
| $1C_H$ | ATOM1_OUT_0 | √ | √ |
| $1D_H$ | ATOM1_OUT_0_N | √ | √ |
| $1E_H$ | ATOM1_OUT_1 | √ | √ |
| $1F_H$ | ATOM1_OUT_1_N | √ | √ |
| $20_H$ | ATOM1_OUT_2 | √ | √ |
| $21_H$ | ATOM1_OUT_2_N | √ | √ |
| $22_H$ | ATOM1_OUT_3 | √ | √ |
| $23_H$ | ATOM1_OUT_3_N | √ | √ |
| $24_H$ | ATOM1_OUT_4 | — | √ |
| $25_H$ | ATOM1_OUT_5 | — | √ |
| $26_H$ | ATOM1_OUT_6 | — | √ |
| $27_H$ | ATOM1_OUT_7 | — | √ |
| $28_H$ | ATOM2_OUT_0 | — | √ |
| $29_H$ | ATOM2_OUT_0_N | — | √ |
| $2A_H$ | ATOM2_OUT_1 | — | √ |
| $2B_H$ | ATOM2_OUT_1_N | — | √ |
| $2C_H$ | ATOM2_OUT_2 | — | √ |
| $2D_H$ | ATOM2_OUT_2_N | — | √ |
| $2E_H$ | ATOM2_OUT_3 | — | √ |
| $2F_H$ | ATOM2_OUT_3_N | — | √ |
| $30_H$ | ATOM2_OUT_4 | — | √ |
| $31_H$ | MCS0_IRQ_0 | √ | √ |
| $32_H$ | MCS0_IRQ_1 | √ | √ |
| $33_H$ | MCS0_IRQ_2 | √ | √ |
| $34_H$ | MCS0_IRQ_3 | √ | √ |
| $35_H$ | MCS0_IRQ_4 | √ | √ |
| $36_H$ | MCS0_IRQ_5 | √ | √ |

**Table 26.8 GTM trigger source selection by PREMUXnx bits; x means scan group: 0 to 4 (3/3)**

| PREMUXnx | Which GTM trigger source selected for ADCF SGx | Supporting device | |
|---|---|---|---|
| | | P1M-C | P1H-C, P1H-CE |
| $37_H$ | MCS0_IRQ_6 | √ | √ |
| $38_H$ | MCS0_IRQ_7 | √ | √ |
| $39_H$ | MCS0_IRQ_8 | √ | √ |
| $3A_H$ | MCS1_IRQ_0 | — | √ |
| $3B_H$ | MCS1_IRQ_1 | — | √ |
| $3C_H$ | MCS1_IRQ_2 | — | √ |
| $3D_H$ | MCS1_IRQ_3 | — | √ |
| $3E_H$ | MCS1_IRQ_4 | — | √ |
| $3F_H$ | MCS1_IRQ_5 | — | √ |

**Note:** If the selection code of PREMUX indicates unused signal such as PREMUXnx=100100 for device P1M-C, the result of PREMUX selection is 0.

### 26.3.3 PIC2CTRGMUXn — A/D converter trigger select control n (n = 0/1)

The PIC2CTRGMUX0 and PIC2CTRGMUX1 registers select the trigger of ADCF external pin or GTM trigger respectively for ADCF0 and ADCF1.

**Access:** This register can be read/written in 32-bit units

**Address:** PIC2CTRGMUX0 <PIC2C_base> + $10_H$
PIC2CTRGMUX1 <PIC2C_base> + $14_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MUXn4 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | MUXn3 | — | — | — | MUXn2 | — | — | — | MUXn1 | — | — | — | MUXn0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R | R | R | R/W | R | R | R | R/W | R | R | R | R/W |

**Table 26.9 PIC2CTRGMUXn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | When read, the value after reset is read. Writing is ignored. |
| 16 | MUXn4 | Selects ADCnTRG or GTM trigger for ADCFn scan group 4<br>0: ADCnTRG<br>1: Signal selected by PIC2CTRGPREMUXn.PREMUXn4 bits |
| 15 to 13 | Reserved | When read, the value after reset is read. Writing is ignored. |
| 12 | MUXn3 | Selects ADCnTRG or GTM trigger for ADCFn scan group 3<br>0: ADCnTRG<br>1: Signal selected by PIC2CTRGPREMUXn.PREMUXn3 bits |
| 11 to 9 | Reserved | When read, the value after reset is read. Writing is ignored. |
| 8 | MUXn2 | Selects ADCnTRG or GTM trigger for ADCFn scan group 2<br>0: ADCnTRG<br>1: Signal selected by PIC2CTRGPREMUXn.PREMUXn2 bits |
| 7 to 5 | Reserved | When read, the value after reset is read. Writing is ignored. |
| 4 | MUXn1 | Selects ADCnTRG or GTM trigger for ADCFn scan group 1<br>0: ADCnTRG<br>1: Signal selected by PIC2CTRGPREMUXn.PREMUXn1 bits |
| 3 to 1 | Reserved | When read, the value after reset is read. Writing is ignored. |
| 0 | MUXn0 | Selects ADCnTRG or GTM trigger for ADCFn scan group 0<br>0: ADCnTRG<br>1: Signal selected by PIC2CTRGPREMUXn.PREMUXn0 bits |

### 26.3.4 PIC2CEDGSELn — A/D converter trigger edge control n (n = 0/1)

The PIC2CEDGSEL0 and PIC2CEDGSEL1 registers select active edge of one-shot pulse generator respectively for ADCF0 and ADCF1 triggers.

**Access:** This register can be read/written in 32-bit units

**Address:** PIC2CEDGSEL0 <PIC2C_base> + $18_H$
PIC2CEDGSEL1 <PIC2C_base> + $1C_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | EDGSELn4 | | EDGSELn3 | | EDGSELn2 | | EDGSELn1 | | EDGSELn0 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 26.10 PIC2CEDGSELn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 10 | Reserved | When read, the value after reset is read. Writing is ignored. |
| 9, 8 | EDGSELn4 | Select active edge on ADCFn scan group 4<br>00: Rising edge<br>01: Falling edge<br>10: Both edges<br>11: Setting prohibited |
| 7, 6 | EDGSELn3 | Select active edge on ADCFn scan group 3<br>00: Rising edge<br>01: Falling edge<br>10: Both edges<br>11: Setting prohibited |
| 5, 4 | EDGSELn2 | Select active edge on ADCFn scan group 2<br>00: Rising edge<br>01: Falling edge<br>10: Both edges<br>11: Setting prohibited |
| 3, 2 | EDGSELn1 | Select active edge on ADCFn scan group 1<br>00: Rising edge<br>01: Falling edge<br>10: Both edges<br>11: Setting prohibited |
| 1, 0 | EDGSELn0 | Select active edge on ADCFn scan group 0<br>00: Rising edge<br>01: Falling edge<br>10: Both edges<br>11: Setting prohibited |

### 26.3.5 PIC2CENP2TIM0 — Path to TIM0 enable control register

The PIC2CENP2TIM0 register controls the multiplexer routing the signals from GTM external pin, ADCF interrupt, and RLIN3 external pin to GTM.TIM0.

**Access:** This register can be read/written in 32-bit units

**Address:** <PIC2C_base> + $20_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENP2TIM07 | | ENP2TIM06 | | ENP2TIM05 | | ENP2TIM04 | | ENP2TIM03 | | ENP2TIM02 | | ENP2TIM01 | | ENP2TIM00 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 26.11 PIC2CENP2TIM0 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. Writing is ignored. |
| 15, 14 | ENP2TIM07 | Selects input signal for TIM0_7<br>00: GTM0I7<br>01: Off<br>10: Off<br>11: Off |
| 13, 12 | ENP2TIM06 | Selects input signal for TIM0_6<br>00: GTM0I6<br>01: RLIN31RX<br>10: Off<br>11: Off |
| 11, 10 | ENP2TIM05 | Selects input signal for TIM0_5<br>00: GTM0I5<br>01: RLIN30RX<br>10: Off<br>11: Off |
| 9, 8 | ENP2TIM04 | Selects input signal for TIM0_4<br>00: GTM0I4<br>01: ADCF0 SG4 interrupt<br>10: Off<br>11: Off |
| 7, 6 | ENP2TIM03 | Selects input signal for TIM0_3<br>00: GTM0I3<br>01: ADCF0 SG3 interrupt<br>10: Off<br>11: Off |
| 5, 4 | ENP2TIM02 | Selects input signal for TIM0_2<br>00: GTM0I2<br>01: ADCF0 SG2 interrupt<br>10: Off<br>11: Off |
| 3, 2 | ENP2TIM01 | Selects input signal for TIM0_1<br>00: GTM0I1<br>01: ADCF0 SG1 interrupt<br>10: Off<br>11: Off |

**Table 26.11 PIC2CENP2TIM0 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1, 0 | ENP2TIM00 | Selects input signal for TIM0_0<br>00: GTM0I0<br>01: ADCF0 SG0 interrupt<br>10: Off<br>11: Off |

**Note:** The statement “Off” means 0 data.

### 26.3.6 PIC2CENP2TIM1 — Path to TIM1 enable control register

The PIC2CENP2TIM1 register controls the multiplexer routing the signals from GTM external pin, ADCF interrupt, and RLIN3 external pin to GTM.TIM1.

**Access:** This register can be read/written in 32-bit units

**Address:** <PIC2C_base> + $24_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENP2TIM17 | | ENP2TIM16 | | ENP2TIM15 | | ENP2TIM14 | | ENP2TIM13 | | ENP2TIM12 | | ENP2TIM11 | | ENP2TIM10 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 26.12 PIC2CENP2TIM1 Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. Writing is ignored. |
| 15, 14 | ENP2TIM17 | Selects input signal for TIM1_7<br>00: GTM1I7<br>01: Off<br>10: Off<br>11: Off |
| 13, 12 | ENP2TIM16 | Selects input signal for TIM1_6<br>00: GTM1I6<br>01: RLIN33RX (Available only in P1H-C, P1H-CE; otherwise off)<br>10: Off<br>11: Off |
| 11, 10 | ENP2TIM15 | Selects input signal for TIM1_5<br>00: GTM1I5<br>01: RLIN32RX (Available only in P1H-C, P1H-CE; otherwise off)<br>10: Off<br>11: Off |
| 9, 8 | ENP2TIM14 | Selects input signal for TIM1_4<br>00: GTM1I4<br>01: ADCF0 SG4 interrupt<br>10: ADCF1 SG4 interrupt<br>11: Off |
| 7, 6 | ENP2TIM13 | Selects input signal for TIM1_3<br>00: GTM1I3<br>01: ADCF0 SG3 interrupt<br>10: ADCF1 SG3 interrupt<br>11: Off |
| 5, 4 | ENP2TIM12 | Selects input signal for TIM1_2<br>00: GTM1I2<br>01: ADCF0 SG2 interrupt<br>10: ADCF1 SG2 interrupt<br>11: Off |
| 3, 2 | ENP2TIM11 | Selects input signal for TIM1_1<br>00: GTM1I1<br>01: ADCF0 SG1 interrupt<br>10: ADCF1 SG1 interrupt<br>11: Off |

**Table 26.12 PIC2CENP2TIM1 Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1, 0 | ENP2TIM10 | Selects input signal for TIM1_0<br>00: GTM1I0<br>01: ADCF0 SG0 interrupt<br>10: ADCF1 SG0 interrupt<br>11: Off |

**Note:** The statement “Off” means 0 data.

### 26.3.7 PIC2CENHIZDTM — Hi-Z function for GTM output enable control register

The PIC2CENHIZDTM register enables Hi-Z function for GTM outputs.

**Access:** This register can be read/written in 32-bit units

**Address:** <PIC2C_base> + $28_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | HIZ2ECM | HIZ1ECM | HIZ0ECM | HIZ2ESO | HIZ1ESO | HIZ0ESO |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 26.13 PIC2CENHIZDTM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. Writing is ignored. |
| 5 | HIZ2ECM | Enables/disables Hi-Z control by ECM for GTM ATOM2 outputs<br>0: Disabled<br>1: Enabled |
| 4 | HIZ1ECM | Enables/disables Hi-Z control by ECM for GTM ATOM1 outputs<br>0: Disabled<br>1: Enabled |
| 3 | HIZ0ECM | Enables/disables Hi-Z control by ECM for GTM ATOM0 outputs<br>0: Disabled<br>1: Enabled |
| 2 | HIZ2ESO | Enables/disables Hi-Z control by ESO2Z for GTM ATOM2 outputs<br>0: Disabled<br>1: Enabled |
| 1 | HIZ1ESO | Enables/disables Hi-Z control by ESO1Z for GTM ATOM1 outputs<br>0: Disabled<br>1: Enabled |
| 0 | HIZ0ESO | Enables/disables Hi-Z control by ESO0Z for GTM ATOM0 outputs<br>0: Disabled<br>1: Enabled |

## 26.4 Operation

### 26.4.1 ADCF trigger select function

Each ADCF is equipped with five scan groups. For every scan group there is a hardware trigger signal.

To increase the amount of possible trigger sources the PIC implements a multiplexer structure which allows the selection of different trigger signals. This function is built with basically two multiplexers per scan group. The first multiplexer makes a pre-selection of the GTM output signals so that only the selected signal is routed over the chip to the input of the second multiplexer.

The second multiplexer is connected to the trigger input of the ADCF. The input signals for the second multiplexer come from the GTM and one comes from a port (ADCnTRG).

The configurable edge detectors are implemented in the output of multiplexers. The edge detectors can be configured to detect rising, falling or both edges of an incoming signal.

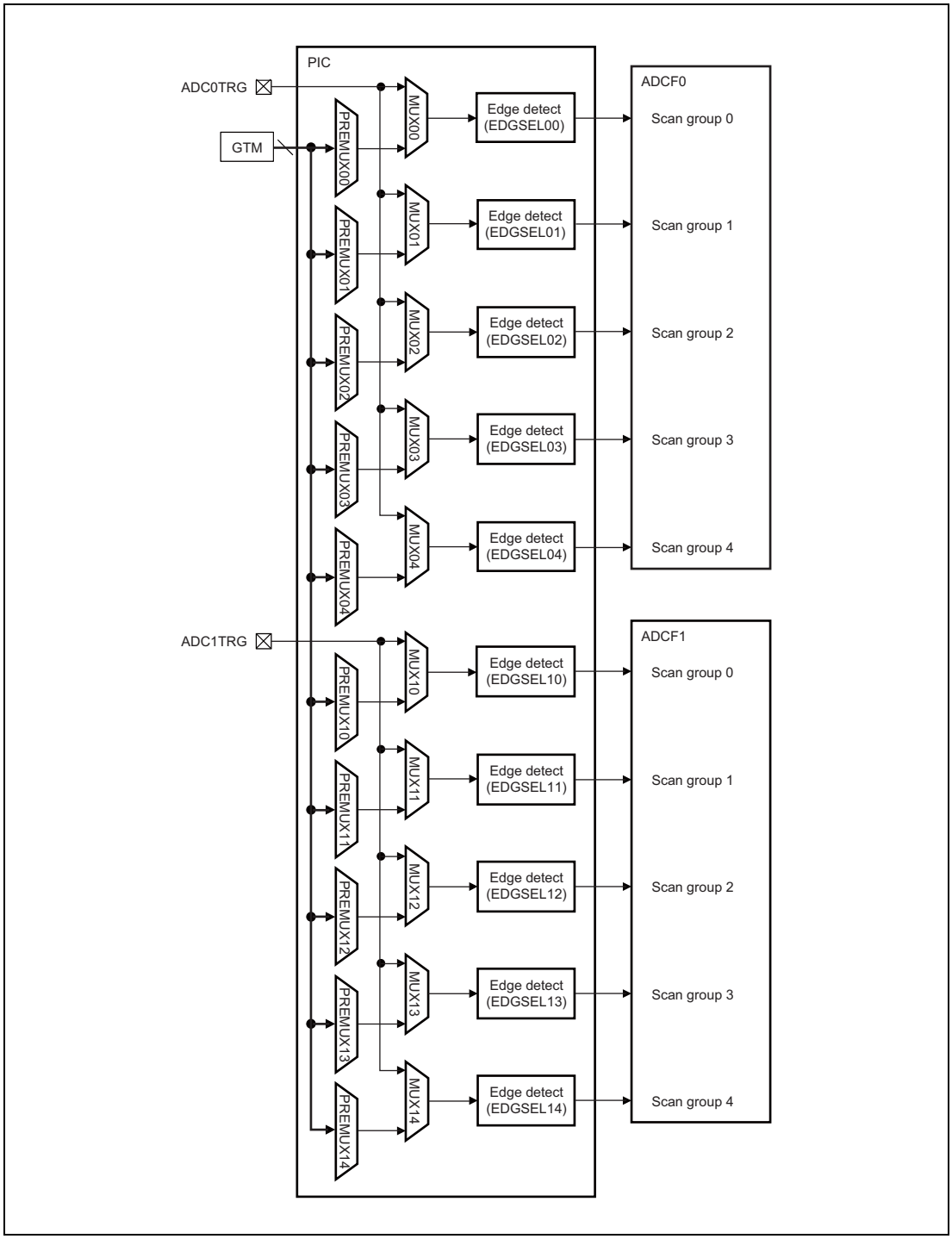



**Figure 26.2 Connection from GTM trigger sources to ADCF trigger inputs**

### 26.4.2 ADCF conversion interrupt routed to GTM input

Each ADCF channel group has an interrupt signalizing that the ADCF conversion has ended. This interrupt can be multiplexed to the GTM inputs.

**Table 26.14 Possible connection of ADCF interrupts to GTM inputs**

| Interrupt source | GTM input |
|---|---|
| ADCF0 IRQ conversion group 0 | TIM0_0, TIM1_0 |
| ADCF0 IRQ conversion group 1 | TIM0_1, TIM1_1 |
| ADCF0 IRQ conversion group 2 | TIM0_2, TIM1_2 |
| ADCF0 IRQ conversion group 3 | TIM0_3, TIM1_3 |
| ADCF0 IRQ conversion group 4 | TIM0_4, TIM1_4 |
| ADCF1 IRQ conversion group 0 | TIM1_0 |
| ADCF1 IRQ conversion group 1 | TIM1_1 |
| ADCF1 IRQ conversion group 2 | TIM1_2 |
| ADCF1 IRQ conversion group 3 | TIM1_3 |
| ADCF1 IRQ conversion group 4 | TIM1_4 |

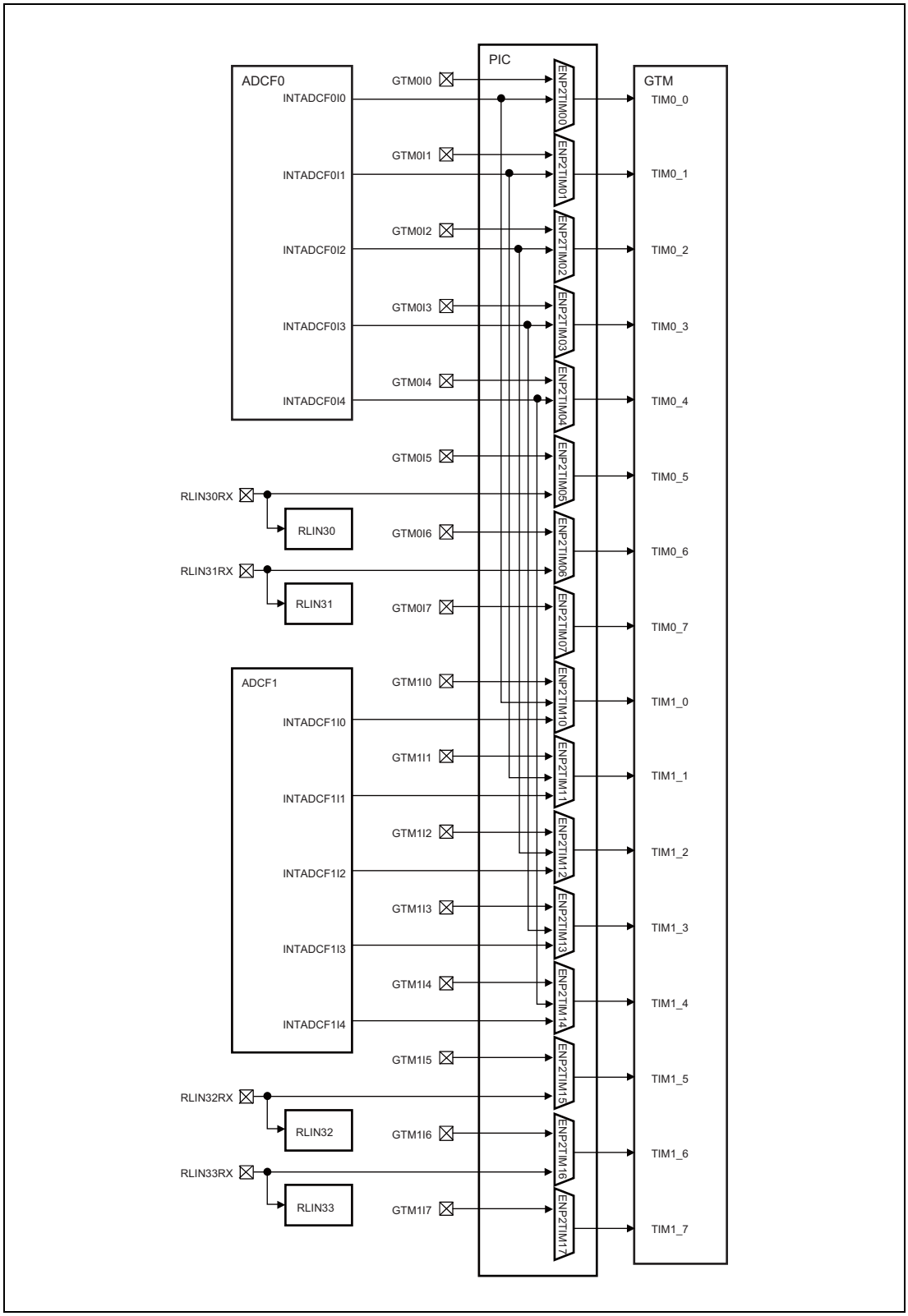



**Figure 26.3 Connections for GTM inputs from external pin, ADCF, and RLIN3**

### 26.4.3 Baud rate measurement for an UART (RLIN3)

To measure the baud rate of the received data a connection from the RLIN3 soft macro data input (RX) to GTM input is made. The connection can be selected by a control register.

**Table 26.15 Connections between RLIN3 and GTM**

| Signal source | GTM input | Supporting device | |
|---|---|---|---|
| | | P1M-C, P1H-C (4MB, BGA-156) | P1H-C (4MB, BGA-292), P1H-C (8MB), P1H-CE |
| RLIN30RX | TIM0_5 | √ | √ |
| RLIN31RX | TIM0_6 | √ | √ |
| RLIN32RX | TIM1_5 | — | √ |
| RLIN33RX | TIM1_6 | — | √ |

### 26.4.4 Hi-Z control function over external pin for GTM output

The I/O driven by GTM output can be set to Hi-Z over an external pin but also by the ECM module within 50ns. The path to Hi-Z control of the I/O buffers can be enabled by a register in the PIC module. The signals from the ECM and from the ESO pin can be masked independently from each other.

For this device, GTM has three output groups and each group has an independent Hi-Z control signal. The controlled output signals are GTMATnOm and GTMATnOmN with n = 0 to 2 and m = 0 to 3.

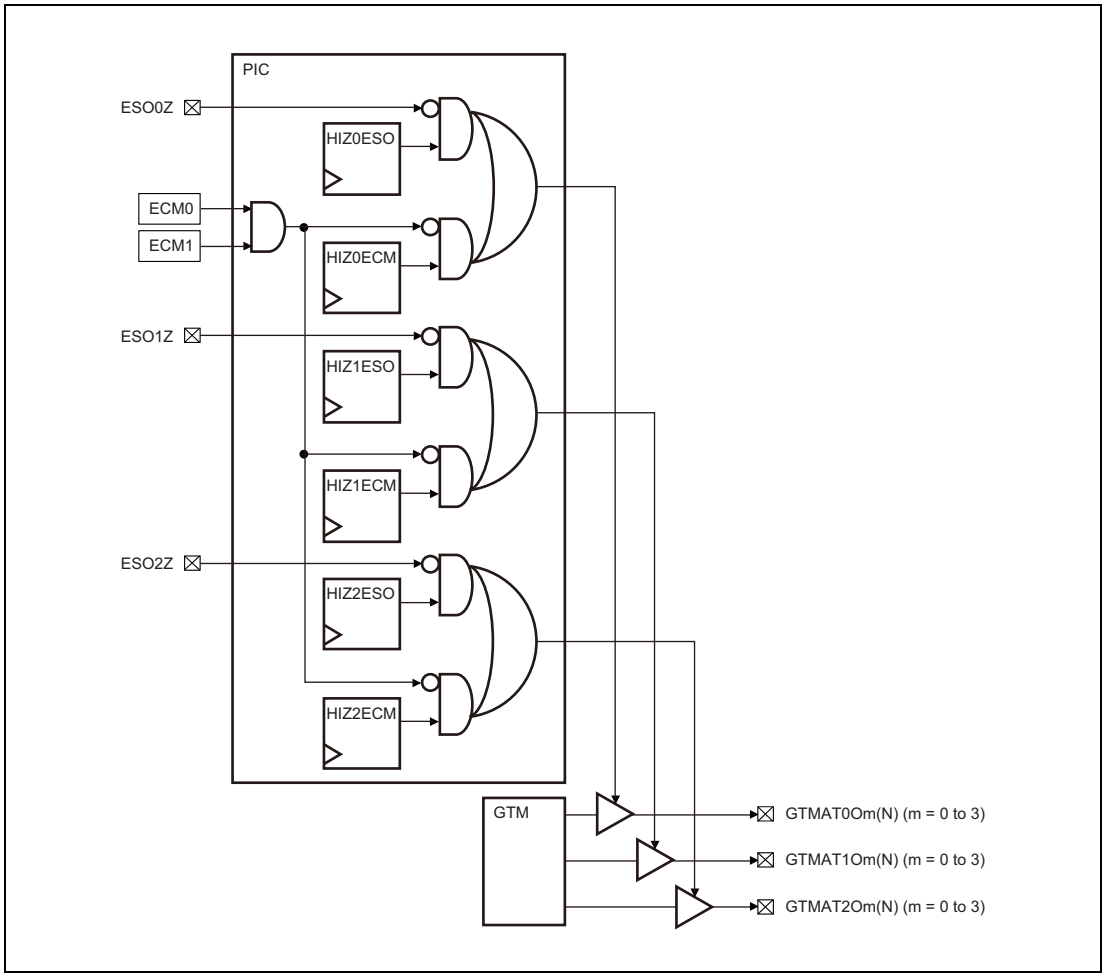



**Figure 26.4 Hi-Z control of GTM output**

**NOTE**

In P1M-C, the input from ECM1 is clamped to non-active state, that is high, because of ECM1 not existing.

### 26.4.5 GTM output monitor for PWM diagnostic

This function helps to verify the output of the GTM by loop-back. The monitoring point is in the IO-buffer itself were the input path must be enabled over the port function. The signal is routed back to GTM input over the PIC. In the PIC this path can be enabled over a register. The block diagram is shown in **Figure 26.5**.

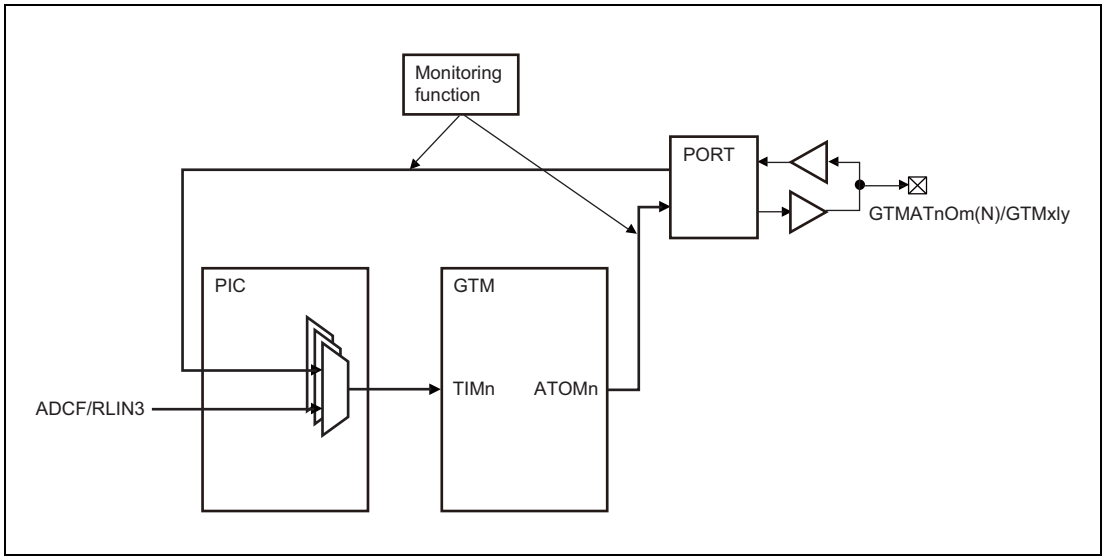



**Figure 26.5 Block diagram for GTM output monitoring function**

The assignment of the GTM outputs and the corresponding GTM inputs is bound to the pin multiplexing since the monitoring point is the pin itself. For details of the assignment, see "**Section 2, Pin Functions**".

**NOTE**

For this function the respective port has to be set into bi-directional mode and TIMn inputs have to be configured accordingly.

## 26.5 Difference among P1M-C, P1H-C and P1H-CE

**Table 26.16** shows each device different specification.

**Table 26.16 Each device different specification**

| Specification | P1M-C | P1H-C, P1H-CE |
|---|---|---|
| Number of GTM trigger sources | ATOM0: 12<br>ATOM1: 8<br>ATOM2: —<br>TIM0: 8<br>TIM1: 8<br>MCS0: 9<br>MCS1: —<br>(Total: 45) | ATOM0: 12<br>ATOM1: 12<br>ATOM2: 9<br>TIM0: 8<br>TIM1: 8<br>MCS0: 9<br>MCS1: 6<br>(Total: 64) |
| Number of RLIN3 trigger sources | RLIN3: 2 | RLIN3: 4*1 |
| Number of groups for Hi-Z control | ESOnZ: 2<br>(n = 0 to 1) | ESOnZ: 3*1<br>(n = 0 to 2) |

Note 1. P1H-C (4MB, BGA-156) does not support RLIN32-33 and ESO2Z.

# Section 27 A/D Converter (ADCF)

This section contains a generic description of the A/D converter (ADCF).

The first part of the section describes all RH850/P1x-C specific properties such as the number of units, register base address, etc.

The remainder of the section describes the function of ADCF and registers.

## 27.1 Features of RH850/P1x-C ADCF

### 27.1.1 Number of Units

This LSI has the following number of units of ADCF.

**Table 27.1 Units**

| Products | P1M-C, P1H-C, P1H-CE |
|---|---|
| Number of units | 2 |
| Name | ADCFn (n = 0, 1) |

**Table 27.2 Index**

| Index | Meaning |
|---|---|
| n | Throughout this section, the individual ADCF units are identified by the index “n” (n = 0, 1), for example, ADCFnVCRj for the virtual channel register n. |
| m | Throughout this section, the number of each ADCF channel is indicated by the index “m”. This index is difference among products. (See **Table 27.44**) |
| j | Throughout this section, the number of data registers and virtual channels are identified by the index “j”, for example, ADCFnDRj for the data register j . This index is difference among products. (See **Table 27.44**) |
| x | Throughout this section, the scan group is indicated by the letter “x” (x = 0 to 4). |
| y | Throughout this section, the number of A/D timers are indicated by the letter “y” (y = 3, 4). |

### 27.1.2 Register Base Address

ADCF base addresses are listed in the following table.

ADCF register addresses are given as offsets from the individual base address.

**Table 27.3 Register Base Address**

| Base Address Name | Base Address |
|---|---|
| <ADCF0_base> | FFF9 $1000_H$ |
| <ADCF1_base> | FFF9 $2000_H$ |

### 27.1.3 Clock Supply

Clock supply to ADCF is listed in the following table.

**Table 27.4 Clock Supply**

| Unit Name | Supply Clock Name |
|---|---|
| ADCFn | Low-speed system clock (CLK_LSB) |

**Note:** Only when CLK_LSB = 8 MHz – 40 MHz, the electrical characteristics can be guranteed.

### 27.1.4 Interrupts and DMA

ADCF interrupt requests are listed in the following table.

**Table 27.5 Interrupt Requests**

| Interrupt name | Outline | Interrupt Number | DMA Trigger Number | DTS Trigger Number (Primary) |
|---|---|---|---|---|
| **ADCF0** | | | | |
| INTADCF0I0 | ADCF0 scan group 0 end interrupt | 81 | 66 | 26 |
| INTADCF0I1 | ADCF0 scan group 1 end interrupt | 82 | 67 | 27 |
| INTADCF0I2 | ADCF0 scan group 2 end interrupt | 83 | 68 | 28 |
| INTADCF0I3 | ADCF0 scan group 3 end interrupt | 84 | 69 | 29 |
| INTADCF0I4 | ADCF0 scan group 4 end interrupt | 85 | 70 | 30 |
| INTADCF0ERR | ADCF0 AD error interrupt | 80 | — | — |
| ADMPXI0 | ADCF0 MPX DMA trigger request | — | 71 | — |
| **ADCF1** | | | | |
| INTADCF1I0 | ADCF1 scan group 0 end interrupt | 167 | 72 | 31 |
| INTADCF1I1 | ADCF1 scan group 1 end interrupt | 168 | 73 | 32 |
| INTADCF1I2 | ADCF1 scan group 2 end interrupt | 169 | 74 | 33 |
| INTADCF1I3 | ADCF1 scan group 3 end interrupt | 170 | 75 | 34 |
| INTADCF1I4 | ADCF1 scan group 4 end interrupt | 171 | 76 | 35 |
| INTADCF1ERR | ADCF1 AD error interrupt | 166 | — | — |
| ADMPXI1 | ADCF1 MPX DMA trigger request | — | 77 | — |

**Note:** ADC parity error interrupt is connected to ECM (error control module).

The Error interrupt is logical sum of the following interrupt.

- Upper/Lower Limit compare error
- Overwrite error
- ID error

### 27.1.5 Hardware Reset

ADCF reset sources are listed in the following table. ADCF is initialized by these reset condition.

**Table 27.6 Reset Condition**

| Reset Name | Reset condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| All Registers | √ | √ | √ | √ | √ |

### 27.1.6 External Input/Output Signals

External input/output signals of ADCF are listed in the following table.

**Table 27.7 External Input/Output Signals**

| Pin Name Definition in this section | Pin Name Definition in RH850/P1x-C | Outline |
|---|---|---|
| A0Vcc | A0VCC | Power supply pin for the input analog part |
| A0Vss | A0VSS | Ground pin for the input analog part |
| A1Vcc | A1VCC | Power supply pin for the input analog part |
| A1Vss | A1VSS | Ground pin for the input analog part |
| A0VREFH | A0VREFH | Reference voltage pin for the input analog part |
| A1VREFH | A1VREFH | Reference voltage pin for the input analog part |
| A0VREFL | — | Reference voltage pin for the input analog part (Merged A0VSS) |
| A1VREFL | — | Reference voltage pin for the input analog part (Merged A1VSS) |
| AN0Im | ADC0Im | ADCF0 12-bit resolution analog input pin, physical channel m. |
| AN1Im | ADC1Im | ADCF1 12-bit resolution analog input pin, physical channel m. |
| ADTRG0 | ADC0TRG | External input trigger pin |
| ADTRG1 | ADC1TRG | External input trigger pin |
| ADEND0 | ADC0CNV | A/D conversion timing monitor pin |
| ADEND1 | ADC1CNV | A/D conversion timing monitor pin |

Please substitute “Pin Name Definition in RH850/P1x-C” for “Pin Name Definition in this section” when using RH850/P1x-C.

The input channel for the external analog multiplexer (MPX) is fixed to one dedicated physical input channel and is different for the implemented ADCFn modules.

- The physical input channel for MPX for ADCF0 is AN0I06
- The physical input channel for MPX for ADCF1 is AN1I00

### 27.1.7 Analog Channels

A/D conversion is available for ADCF0 and ADCF1. For corresponding channel number for each product, see **Section 27.7, Difference among P1M-C, P1H-C and P1H-CE**.

### 27.1.8 Virtual Channel

Each ADCF has virtual channels shown **Table 27.44**. Analog channels for which A/D conversion is to be made and other accompanying information are set for each virtual channel. By sequentially performing the processing for the virtual channels indicated by the start virtual channel pointer and the end virtual channel pointer in each scan group, scans (which can perform A/D conversion for any analog channels in any order) can be executed.

## 27.2 Overview

### 27.2.1 Functional Overview

ADCF has the following features.

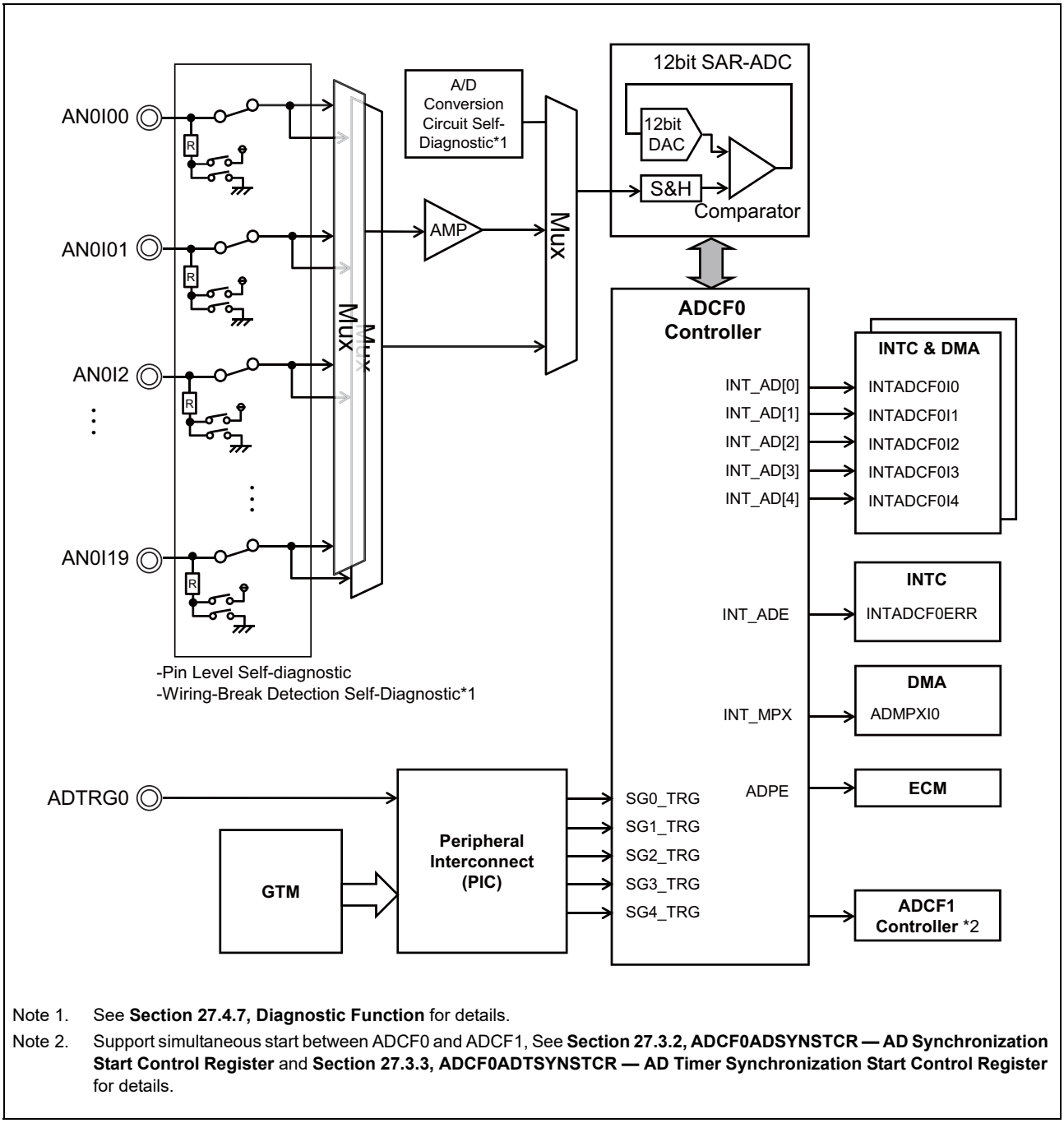
- Advanced A/D converter
  Resolution: 12 bits
  A/D conversion method: Successive approximation method
  Conversion speed: 1.0µs
- Supporting five scan groups
  Each ADCF has five scan groups. Scan settings can be made independently for each scan group.
- Two scan modes
  Each ADCF has two scan modes.
  Multicycle scan mode: Specified number of scans are executed.
  Continuous scan mode: Scans are repeatedly executed without limit.
- Interval function
  The ADCF can start scan groups in any cycle by using the A/D timer equipped in the scan groups 3 and 4. This enables scans with intervals inserted.
- A/D-converted value adding function
  The ADCF performs A/D conversion sequentially twice or four times for a channel, and stores the addition result in the data register. The addition count can be set for each virtual channel.
  The effect of the moving average filter can be gained by using this result. However, this function does not always ensure that A/D conversion accuracy is improved.
- Extended physical channels
  Each ADCF can extend physical channels by using an external analog multiplexer. (Available channel are AN0I06 and AN1I00.)
- Virtual channel concept
  Number of virtual channels larger than number of physical channels.
  Each virtual channels can freely be assigned to each physical conversion channels.
- Data registers
  Data registers corresponding to virtual channels are provided.
- Start trigger for each scan group
  Hardware triggers and software triggers can start processing of each scan group. Only scan groups 3 and 4 can start processing by an A/D timer trigger.
- Asynchronous/synchronous suspend and resume function
  A processing for a scan group can interrupt an ongoing processing for another scan group. The priority is as follows:
  Low High
  SG0 < SG1 < SG2 < SG3 < SG4 (SG: Scan group)
  If a request for a higher-priority SG is present while a lower-priority SG is being processed, the lower-priority SG is suspended after the ongoing virtual channel processing is stopped (synchronous suspend) or after the ongoing virtual channel processing is immediately stopped (asynchronous suspend), and then the processing for the higher-priority SG is performed. After the processing for the higher-priority SG is completed, the suspended virtual channel processing of the lower-priority SG resumes

Also, you can set as follows: when a higher-priority SG interrupts an SG0 processing, asynchronous suspend occurs, but when a higher-priority SG interrupts a lower-priority SG other than SG0, synchronous suspend occurs.

- Supporting scan end interrupt and DMA transfer
  Each scan group can generate an interrupt request to the INTC and activate the DMAC each time a processing for the virtual channel indicated by the end virtual channel pointer ends or a virtual channel ends.
- An analog conversion voltage settable
  The A0VREFH pin and the A1VREFH pin can be used to set the voltage range for analog conversion.
- Abundant safety functions
  The ADCF is equipped with abundant safety functions, including A/D conversion circuit diagnostic function, pin-level diagnostic function, wiring-break detection, normality check for analog selection, upper-limit/lower-limit check for data registers, parity check for data registers, overwrite check for data registers, and read and clear function for data registers.

### 27.2.2 Block Diagram

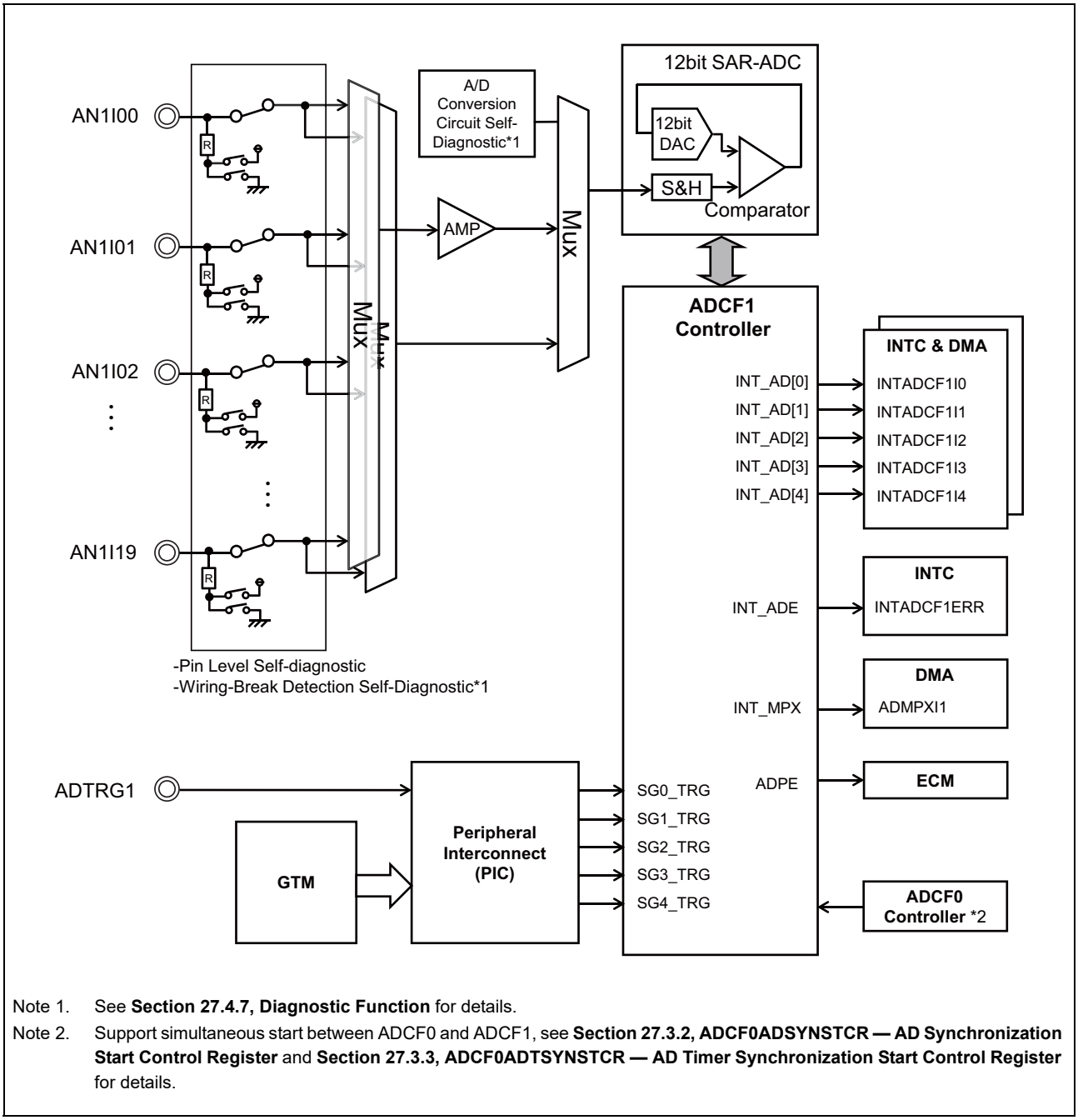
**Figure 27.1** and **Figure 27.2** shows the block diagrams of ADCF.




Note 1. See **Section 27.4.7, Diagnostic Function** for details.

Note 2. Support simultaneous start between ADCF0 and ADCF1, See **Section 27.3.2, ADCF0ADSYNSTCR — AD Synchronization Start Control Register** and **Section 27.3.3, ADCF0ADTSYNSTCR — AD Timer Synchronization Start Control Register** for details.

**Figure 27.1 Block Diagram (CH0)**




Note 1. See **Section 27.4.7, Diagnostic Function** for details.

Note 2. Support simultaneous start between ADCF0 and ADCF1, see **Section 27.3.2, ADCF0ADSYNSTCR — AD Synchronization Start Control Register** and **Section 27.3.3, ADCF0ADTSYNSTCR — AD Timer Synchronization Start Control Register** for details.

**Figure 27.2 Block Diagram (CH1)**

### 27.2.3 Physical Channels, Virtual Channels and Scan Groups

Each physical channel m refers to an external A/D Converter input ANnIm.
A virtual channel is the A/D Converter channel from the application software's perspective.
The number j of virtual channels is larger than the number of physical channels m.
Each virtual channel j can be assigned to each physical channel via the GCTRL[4:0] bits of its virtual channel register ADCFnVCRj.

An arbitrary number of consecutive virtual channels can be combined as one of the five scan groups SGx, with x = 0 to 4.

The set of virtual channels of a scan group is defined by the

- scan group x start virtual channel pointer ADCFnSGVCSPx.VCSP[5:0]
- scan group x end virtual channel pointer ADCFnSGVCEPx.VCEP[5:0]

The conversion results of the virtual channels are stored in the ADCFnDRj registers.
Each 32-bit ADCFnDRj register stores the result of two consecutive virtual channels j and (j+1).
The conversion results is also stored in the data supplementary information register ADCFnDIRj (j = 0 to 35). This register contains some additional information about the conversion.

The figure below shows an example with the following configuration:

- Scan group x virtual channels: j = 4 to 9
- Virtual to physical channel assignment:

| Register setting | Virtual channel j | Physical channel m |
|---|---|---|
| ADCFnVCR4.GCTRL[4:0] = 1 | 4 | ANnI01 |
| ADCFnVCR5.GCTRL[4:0] = 3 | 5 | ANnI03 |
| ADCFnVCR6.GCTRL[4:0] = 3 | 6 | ANnI03 |
| ADCFnVCR7.GCTRL[4:0] = 9 | 7 | ANnI09 |
| ADCFnVCR8.GCTRL[4:0] = 8 | 8 | ANnI08 |
| ADCFnVCR9.GCTRL[4:0] = 6 | 9 | ANnI06 |

After a conversion start trigger for scan group x the first virtual channel ADCFnVCR4, i.e. the assigned physical channel ANnI01, is converted and the result is stored in ADCFnDR4.DR4[15:0].
The next conversion result goes to ADCFnDR4.DR5[15:0].
An internal scan pointer is increment and moves to the next virtual channel.

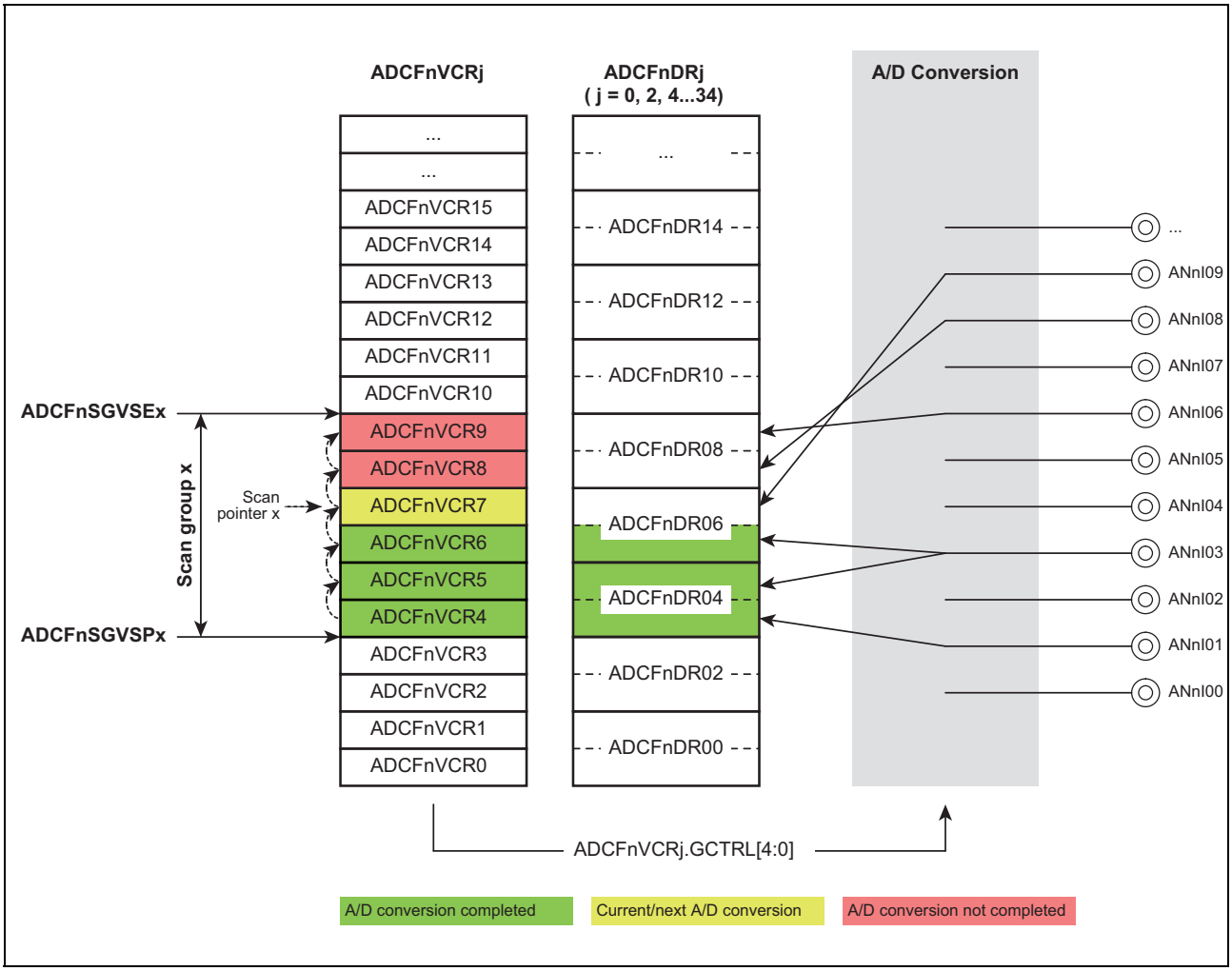



**Figure 27.3 Virtual and Physical Channels**

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

# 27.3 Registers

## 27.3.1 List of Registers

ADCF registers are listed in the following table.

**Table 27.8 List of Registers (1/2)**

| Register Name | Symbol | Address | Access Protection | |
|---|---|---|---|---|
| | | | PBG | Others |
| **ADC common registers** | | | | |
| AD synchronization start control register | ADCF0ADSYNSTCR | <ADCF0_base> + $300_H$ | PBG4#0 .PG4-ADC0 | — |
| AD timer synchronization start control register | ADCF0ADTSYNSTCR | <ADCF0_base> + $304_H$ | PBG4#0 .PG4-ADC0 | — |
| **ADC specific registers (virtual channel)** | | | | |
| Virtual channel register j (j = 0 to 35) | ADCFnVCRj | <ADCFn_base> + j × $4_H$ | *1 | — |
| Data register j (j = 0, 2, 4....34) | ADCFnDRj | <ADCFn_base> + $100_H$ + j × $2_H$ | *1 | — |
| Data supplementary information register j (j = 0 to 35) | ADCFnDIRj | <ADCFn_base> + $200_H$ + j × $4_H$ | *1 | — |
| **ADC specific registers (control)** | | | | |
| AD halt register | ADCFnADHALTR | <ADCFn_base> + $380_H$ | *1 | — |
| AD control register 1 | ADCFnADCR1 | <ADCFn_base> + $384_H$ | *1 | — |
| MPX current control register | ADCFnMPXCURCR | <ADCFn_base> + $388_H$ | *1 | — |
| MPX current register | ADCFnMPXCURR | <ADCFn_base> + $38C_H$ | *1 | — |
| MPX optional wait register | ADCFnMPXOWR | <ADCFn_base> + $390_H$ | *1 | — |
| AD control register 2 | ADCFnADCR2 | <ADCFn_base> + $398_H$ | *1 | — |
| A/D Conversion Monitor Virtual Channel Pointer | ADCFnADENDP0 | <ADCFn_base> + $3A0_H$ | *1 | — |
| **ADC specific registers (safety-related)** | | | | |
| Safety control register | ADCFnSFTCR | <ADCFn_base> + $3C0_H$ | *1 | — |
| Pin level diagnostic control register | ADCFnTDCR | <ADCFn_base> + $3C4_H$ | *1 | — |
| Upper-limit/lower-limit table register 0 | ADCFnULLMTBR0 | <ADCFn_base> + $3CC_H$ | *1 | — |
| Upper-limit/lower-limit table register 1 | ADCFnULLMTBR1 | <ADCFn_base> + $3D0_H$ | *1 | — |
| Upper-limit/lower-limit table register 2 | ADCFnULLMTBR2 | <ADCFn_base> + $3D4_H$ | *1 | — |
| Error clear register | ADCFnECR | <ADCFn_base> + $3D8_H$ | *1 | — |
| Upper-limit/lower-limit error register | ADCFnULER | <ADCFn_base> + $3DC_H$ | *1 | — |
| Overwrite error register | ADCFnOWER | <ADCFn_base> + $3E0_H$ | *1 | — |
| Parity error register | ADCFnPER | <ADCFn_base> + $3E4_H$ | *1 | — |
| ID error register | ADCFnIDER | <ADCFn_base> + $3E8_H$ | *1 | — |
| **Scan group specific registers** | | | | |
| Scan group x start control register | ADCFnSGSTCRx | <ADCFn_base> + x × $80_H$+$480_H$ | *1 | — |
| AD timer y start control register | ADCFnADTSTCRy | <ADCFn_base> + y × $80_H$+$488_H$ | *1 | — |
| AD timer y end control register | ADCFnADTENDCRy | <ADCFn_base> + y × $80_H$+$48C_H$ | *1 | — |
| Scan group x control register | ADCFnSGCRx | <ADCFn_base> + x × $80_H$+$490_H$ | *1 | — |
| Scan group x start virtual channel pointer | ADCFnSGVCSPx | <ADCFn_base> + x × $80_H$+$494_H$ | *1 | — |
| Scan group x end virtual channel pointer | ADCFnSGVCEPx | <ADCFn_base> + x × $80_H$+$498_H$ | *1 | — |
| Scan group x multicycle register | ADCFnSGMCYCRx | <ADCFn_base> + x × $80_H$+$49C_H$ | *1 | — |
| Scan group x status register | ADCFnSGSRx | <ADCFn_base> + x × $80_H$+$4A4_H$ | *1 | — |

Table 27.8 List of Registers (2/2)

| Register Name | Symbol | Address | Access Protection | |
|---|---|---|---|---|
| | | | PBG | Others |
| AD timer initial phase register y | ADCFnADTIPRy | <ADCFn_base> + y × $80_H$+$4A8_H$ | *1 | — |
| AD timer period register y | ADCFnADTPRRy | <ADCFn_base> + y × $80_H$+$4AC_H$ | *1 | — |
| Scan group x upper-limit/lower-limit table select register | ADCFnULLMSRx | <ADCFn_base> + x × $80_H$ + $4B0_H$ | *1 | — |

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

Note 1. In the case of
n = 0 PBG4#0.PG4-ADC0
n = 1 PBG4#0.PG4-ADC1

### 27.3.2 ADCF0ADSYNSTCR — AD Synchronization Start Control Register

ADCF0ADSYNSTCR is an 8-bit write-only register that controls simultaneous start of A/D conversion for each scan group of ADCF0 and ADCF1. The register bits are always read as 0.

**Access:** This register can be written in 8-bit units.
**Address:** <ADCF0_base> + $300_H$
**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ADSTART |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 27.9 ADCF0ADSYNSTCR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to1 | Reserved | When writing, write the value after reset. |
| 0 | ADSTART | Starts A/D conversion for scan groups of ADCF0 and ADCF1.<br>Start condition of SGx of ADCFn:<br>For SGx of ADCFn, 1 is written to ADSTART when SGACT is 0 and ADSTARTE is 1.<br>A/D conversion is started simultaneously for the scan groups (of ADCF0 and ADCF1) for which ADSTARTE has been set to 1. |

### 27.3.3 ADCF0ADTSYNSTCR — AD Timer Synchronization Start Control Register

ADCF0ADTSYNSTCR is an 8-bit write-only register that controls simultaneous start of count operation for each A/D timer of ADCF0 and ADCF1. The register bits are always read as 0.

**Access:** This register can be written in 8-bit units.
**Address:** <ADCF0_base> + $304_H$
**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ADTSTART |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 27.10 ADCF0ADTSYNSTCR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | ADTSTART | Starts count operation of the A/D timer of ADCF0 and ADCF1.<br>Start condition of SGx of ADCFn:<br>For A/D timer x of ADCFn, 1 is written to ADTSTART when ADTACT is 0 and ADTSTARTE is 1.<br>Count operation is started simultaneously for the A/D timers (of ADCF0 and ADCF1) for which ADTST has been set to 1.<br>**Note:** x = 3, 4 |

### 27.3.4 ADCFnVCRj — Virtual Channel Register j

ADCFnVCRj is a 32-bit readable/writable register used for each virtual channel.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ADCFn_base> + j x $4_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PUE | PDE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNVCLS[2:0] | | | — | — | — | — | — | ADIE | — | — | GCTRL[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 27.11 ADCFnVCRj register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 17, 16 | PUE, PDE | Pull-Up/Down resistor<br>00: Pull-Up and Pull-Down resistor disabled<br>01: Pull-Down resistor enabled. The resistor is enabled during the sample time.<br>10: Pull-Up resistor enabled. The resistor is enabled during the sample time.<br>11: Setting prohibited.<br>**NOTE**<br>Set PUE=0 and PDE=0 when pin level diagnostic function enable.<br>For more details, see **Section 27.4.7.3, Wiring-Break Detection Diagnostic Function** |
| 15 to13 | CNVCLS[2:0] | Conversion type<br>$0_H$: Normal A/D conversion<br>$3_H$: Diagnosis<br>$4_H$: Normal A/D conversion of addition mode<br>$5_H$: Normal A/D conversion with the MPX<br>$6_H$: Normal A/D conversion with the MPX of addition mode<br>Other than above: Setting prohibited |
| 12 to 8 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 7 | ADIE | Virtual channel end interrupt enable<br>0: ADInx is not output at virtual channel end of virtual channel j in SGx<br>1: ADInx is output at virtual channel end of virtual channel j in SGx<br>ADIE in ADCFnSGCRx is independent of ADIE in ADCFnVCRj. For details, see **Section 27.4.15, Scan End Interrupt Request**. |
| 6, 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

**Table 27.11 ADCFnVCRj register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 4 to 0 | GCTRL[4:0] | General control<br>Write 0 to the bits which are not used among GCTRL[4:0]. |

| CNVCLS[2:0] | GCTRL[4:0] |
|---|---|
| **$0_H$: Normal A/D conversion** | GCTRL[4:0] : Physical Channel |
| **$3_H$: Diagnosis** | GCTRL[4:0] : Voltage level in diagnosis<br>$00_H$: AnVREFH × 0<br>$04_H$: AnVREFH × 1/4<br>$08_H$: AnVREFH × 1/2<br>$0C_H$: AnVREFH × 3/4<br>$10_H$: AnVREFH × 1<br>Other than above: Setting prohibited |
| **$4_H$: Normal A/D conversion of addition mode** | GCTRL[4:0] : Physical channel<br>*The count specified by ADDNT is applied to the number of additions |
| **$5_H$: Normal A/D conversion with the MPX** | GCTRL[4:0] : MPX channel is set.<br>MPX value to be transferred to the external analog multiplexer is specified.<br>An interrupt request (INTADCFnMPX) or a DMA request is output by GCTRL[4:0] transfer to ADCFnMPXCURR at the start of virtual channel.<br>The MPX value can be transferred to the external analog multiplexer by transferring ADCFnMPXCURR to PyDR or PyMDR of I/O port after an interrupt or start of DMAC.<br>See **Section 27.4.3**, for more detail. |
| **$6_H$ : Normal A/D conversion with the MPX of addition mode** | GCTRL[4:0] : MPX channel is set<br><br>MPX value to be transferred to the external analog multiplexer is specified.<br><br>An interrupt request (INTADCFnMPX) or a DMA request is output by GCTRL[4:0] transfer to ADCFnMPXCURR at the start of virtual channel.<br><br>The MPX value can be transferred to the external analog multiplexer by transferring ADCFnMPXCURR to PyDR or PyMDR of I/O port after an interrupt or start of DMAC.<br><br>See **Section 27.4.3, Example of Operation of External Analog Multiplexer** for details.<br><br>The number of addition times specified by ADDNT is applied. |
| **Other than above : Setting prohibited** | — |

**CAUTION**

To prevent malfunction, perform ADCFnVCRj settings after the following settings and confirmation.

1. Confirm for all scan groups that ADSTARTE = $0_H$
2. Confirm for all scan groups that TRGMD = $0_H$
3. Confirm for all scan groups that SGACT = $0_H$

[Note]

Setting of the ADCFnVCRj.GCTRL[5:0] bits is allowed only after the following confirmation.

1. Confirm for the related scan group that ADSTARTE = $0_H$
2. Confirm for the related scan group that TRGMD = $0_H$
3. Confirm for the related scan group that SGACT = $0_H$

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

### 27.3.5 ADCFnDRj — Data Register j

This register is a 32-bit read-only register, which stores the A/D conversion results corresponding to ADCFnVCRj and ADCFnVCR (j + 1). As the A/D conversion results, the conversion result for ADCFnVCR (j + 1) is stored in the upper 16 bits, and the conversion result for ADCFnVCRj is stored in the lower 16 bits.
ADCFnDRj format depends on the DFMT setting of ADCFnVCRj and the ADDNT setting (when CNVCLS[2:0] = $4_H$, $6_H$). ADCFnDRj is cleared to $0000_H$ when ADCFnDRj or ADCFnDIRj is read while RDCLRE is set to 1.

**Access:** This register can be read only in 32-bit units.

**Address:** <ADCFn_base> + $100_H$ + j × $2_H$

**Value after reset:** $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DR(j+1)[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DRj[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 27.12 ADCFnDRj register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | DR(j+1)[15:0] | These bits are used to store the A/D conversion result data.<br>(The A/D conversion result for the channel set in ADCFnVCR (j + 1) are transferred.) |
| 15 to 0 | DRj[15:0] | These bits are used to store the A/D conversion result data.<br>(The A/D conversion result for the channel set in ADCFnVCRj are transferred.) |

**NOTE**

j = 0, 2, …, 34

**For signed fixed-point format (DFMT = 0)**

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Convert once | S | | | | | | | | | | | | | 0 | 0 | 0 |
| Convert twice | S | | | | | | | | | | | | | | 0 | 0 |
| Convert 4 timers | S | | | | | | | | | | | | | | | 0 |

Position of decimal point

**For signed integer format (DFMT = 1)**

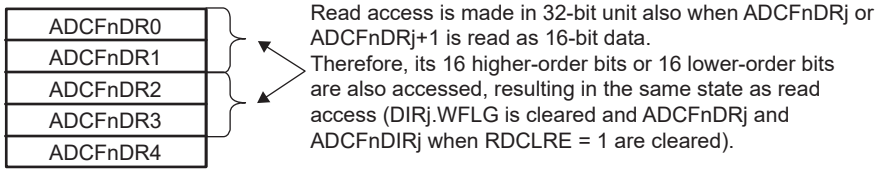
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Convert once | S | S | S | S | | | | | | | | | | | | |
| Convert twice | S | S | S | | | | | | | | | | | | | |
| Convert 4 timers | S | S | | | | | | | | | | | | | | |

Position of decimal point

| | |
|---|---|
| S | : Sign bit (always 0) |
| 0 | : Zero extension |

The format setting in ADDNT is valid when CNVCLS[2:0] = $4_H$, $6_H$.

If CNVCLS[2:0] is not $4_H$, $6_H$, the format is “convert once.”




**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

### 27.3.6 ADCFnDIRj — Data Supplementary Information Register j

ADCFnDIRj is a 32-bit read-only register that stores supplementary information of ADCFnDRj and A/D converted value. ADCFnDIRj is provided for each virtual channel. ADCFnDIRj is cleared to $0000\ 0000_H$ when ADCFnDRj or ADCFnDIRj is read while RDCLRE is set to 1. WFLG is cleared when ADCFnDRj or ADCFnDIRj is read regardless of the RDCLRE setting. This register must always be read as 32-bit data. ADCFnDRj is read from the16 lower-order bits.

**Access:** This register can be read in 32-bit units.

**Address:** <ADCFn_base> + $200_H + j \times 4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | WFLG | PRTY | — | — | — | ID[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DRj | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 27.13 ADCFnDIRj register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is read. |
| 25 | WFLG | Write Flag<br>1: Setting condition<br>An A/D converted value is stored in ADCFnDRj<br>0: Clearing conditions<br>ADCFnDRj or ADCFnDIRj is read. |
| 24 | PRTY | Parity<br>Parity bit (even parity) for ADCFnDRj. |
| 23 to 21 | Reserved | When read, the value after reset is read. |
| 20 to 16 | ID[4:0] | Contains the physical channel information of the converted channel.. |

| CNVCLS[2:0] | GCTRL[4:0] |
|---|---|
| $0_H$: Normal A/D conversion | ID[4:0] : Physical Channel |
| $3_H$: Self-diagnosis | ID[4:0] : Voltage level in self-diagnosis<br>$00_H$: AnVREFH × 0<br>$04_H$: AnVREFH × 1/4<br>$08_H$: AnVREFH × 1/2<br>$0C_H$: AnVREFH × 3/4<br>$10_H$: AnVREFH × 1<br>Other than above : Setting prohibited |
| $4_H$: Normal A/D conversion of addition mode | ID[4:0] : Physical channel |
| $5_H$: Normal A/D conversion with the MPX | ID[4:0] : Physical channel |
| $6_H$ : Normal A/D conversion with the MPX of addition mode | ID[4:0] : Physical channel |
| Other than above : Setting prohibited | — |

| | | |
|---|---|---|
| 15 to 0 | DRj | same as ADCFnDRj |

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

### 27.3.7 ADCFnADHALTR — AD Halt Register

ADCFnADHALTR is an 8-bit write-only register that halts the ADC. The register bits are always read as 0.

**Access:** This register can be written in 8-bit units.

**Address:** <ADCFn_base> + $380_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | HALT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 27.14 ADCFnADHALTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | HALT | Halt<br>All scan groups and AD timers are halted and initialized, and the ADC becomes the idle state.<br>Writing 0: Not halted.<br>Writing 1: Halted. |

### 27.3.8 ADCFnADCR1 — AD Control Register 1

ADCFnADCR1 is an 8-bit readable/writable register for ADC common control.

**Access:** This register can be read/written in 8-bit units.

**Address:** <ADCFn_base> + $384_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | SUSMTD[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 27.15 ADCFnADCR1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | SUSMTD [1:0] | Suspend Method<br>These bits select the suspend method when a higher-priority scan group interrupts a lower-priority scan group.<br>Synchronous suspend: If a request from a higher-priority SG is present while a lower-priority SG is being processed, processing for the lower-priority SG is suspended after the ongoing virtual channel processing is completed, and then processing for the higher-priority SG is executed. After processing for the higher-priority SG is completed, the suspended virtual channel processing for the lower-priority SG is resumed.<br>Asynchronous suspend: If a request from a higher-priority SG is present while a lower-priority SG is being processed, the ongoing virtual channel processing is immediately suspended, and then processing for the higher-priority SG is executed. After processing for the higher-priority SG is completed, the suspended virtual channel processing for the lower-priority SG is resumed.<br>$0_H$: Synchronous suspend<br>$1_H$: Asynchronous suspend when a higher-priority SG interrupts SG0<br>Synchronous suspend when a higher-priority SG interrupts a lower-priority SG (except for SG0)<br>$2_H$: Asynchronous suspend<br>$3_H$: Setting prohibited<br>For the detail, see **Figure 27.8, Example of Synchronous Suspend and Resume Operation** and **Figure 27.9, Example of Asynchronous Suspend and Resume Operation**. |

**CAUTION**

To prevent malfunction, perform ADCFnADCR1 settings after the following settings and confirmation.

(1) ADSTARTE of all scan groups is 0,

(2)TRGMD[0] of scan groups 0,1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$.

(3) SGACT of all scan groups is 0 (before scan groups are started)

### 27.3.9 ADCFnMPXCURCR — MPX Current Control Register

ADCFnMPXCURCR is a register that controls the ADCFnMPXCURR format.

**Access:** This register can be read/written in 8-bit units.

**Address:** <ADCFn_base> + $388_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | MSKCFMT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R/W |

**Table 27.16 ADCFnMPXCURCR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | MSKCFMT0 | MSKC format specification<br>Specifies the MSKC[3:0] format of ADCFnMPXCURR<br>MSKCFMT0<br>0: MSKC[3:0] = 0000<br>1: MSKC[3:0] = 1111 |

**CAUTION**

To prevent malfunction, perform ADCFnMPXCURCR settings after the following settings and confirmation.

(1) ADSTARTE of all scan groups is 0

(2)TRGMD[0] of scan groups 0,1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$

(3) SGACT of all scan groups is 0 (before scan groups are started)

## 27.3.10 ADCFnMPXCURR — MPX Current Register

ADCFnMPXCURR is a 32-bit read-only register that stores the MPX value for an external analog multiplexer.

**Access:** This register can be read in 32-bit units.

**Address:** <ADCFn_base> + $38C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSKC[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MPXCUR[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 27.17 ADCFnMPXCURR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset is read. |
| 19 to 16 | MSKC[3:0] | Mask Control<br>The format depends on the MSKCFMT0 setting of ADCFnMPXCURCR. For details, see **Section 27.3.9, ADCFnMPXCURCR — MPX Current Control Register**. |
| 15 to 4 | Reserved | When read, the value after reset is read. |
| 3 to 0 | MPXCUR[3:0] | Current MPX value<br>When a virtual channel for which CNVCLS[2:0] in ADCFnVCRj is set to $5_H$ or 6H is started, GCTRL[3:0] in ADCFnVCRj is transferred to MPXCUR[3:0].<br>At this time, a DMA transfer request is generated. enabling the MPX value to be sent to an external analog multiplexer.<br>When PSRn is used, transfer the MPX value as a 32-bit value. This enables rewriting only the necessary ports by using the format control in MSKC[3:0]. For details, see **Section 27.4.3, Example of Operation of External Analog Multiplexer**. |

## 27.3.11 ADCFnMPXOWR — MPX Optional Wait Register

ADCFnMPXOWR is a register that specifies the wait time to be inserted for an external analog multiplexer.

**Access:** This register can be read/written in 8-bit units.

**Address:** <ADCFn_base> + $390_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MPXOW[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 27.18 ADCFnMPXOWR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 to 0 | MPXOW | MPX Optional Wait<br>These bits specify the wait time to be inserted before A/D conversion is started after a virtual channel for which CNVCLS[2:0] in ADCFnVCRj is $5_H$ or $6_H$ is started.<br>$0_H$: 0 µs<br>$1_H$: 1 µs<br>$2_H$: 2 µs<br>$3_H$: 3 µs<br>$4_H$: 4 µs<br>$5_H$: 5 µs<br>$6_H$: 6 µs<br>$7_H$: 7 µs<br>$8_H$: 8 µs<br>$9_H$: 9 µs<br>$A_H$: 10 µs<br>$B_H$ to $F_H$: Setting prohibited<br>For details, see **Section 27.4.3, Example of Operation of External Analog Multiplexer**. |

**CAUTION**

To prevent malfunction, perform ADCFnMPXOWR settings after the following settings and confirmation.

(1) ADSTARTE of all scan groups is 0

(2) TRGMD[0] of scan groups 0,1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$.

(3) SGACT of all scan groups is 0 (before scan groups are started).

## 27.3.12 ADCFnADCR2 — AD Control Register 2

ADCFnADCR2 is an 8-bit readable/writable register for ADCF common control.

**Access:** This register can be read/written in 8-bit units.

**Address:** <ADCFn_base> + $398_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | DFMT | — | — | — | ADDNT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R | R | R | R/W |

**Table 27.19 ADCFnADCR2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | DFMT | Data Format<br>0: Signed fixed-point format<br>1: Signed integer format<br>This bit specifies the format of data to be transferred to ADCFnDRj. For details of data format, see **Section 27.3.5, ADCFnDRj — Data Register j**. |
| 3 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | ADDNT | Addition Count Select<br>0: Add twice<br>1: Add 4 times<br>This register is valid only when CNVCLS[2:0] is $4_H$, $6_H$. |

**CAUTION**

To prevent malfunction, perform ADCFnADCR2 settings after the following settings and confirmation.

(1) ADSTARTE of all scan groups is 0

(2) TRGMD[0] of scan groups 0,1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$.

(3) SGACT of all scan groups is 0 (before scan groups are started)

### 27.3.13 ADCFnADENDP0 — A/D Conversion Monitor Virtual Channel Pointer

ADCFnADENDP0 is an 8-bit readable/writable register that selects a virtual channel that outputs the A/D conversion timing to ADENDn.

**Access:** This register can be read/written in 8-bit units.

**Address:** ADCFnADENDP0: <ADCFn_base> + $3A0_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | ENDP[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.20 ADCFnADENDP0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is read. Writing is ignored. |
| 5 to 0 | ENDP[5:0] | A/D Conversion Monitor Virtual Channel Pointer<br>When the virtual channel selected by ADCFnADENDP0 is started, a high level is output to from the ADENDn pin. When the virtual channel selected by ADCFnADENDP0 ends, a low level is output. |

**CAUTION**

To prevent malfunction, perform ADCFnADENDP0 settings after the following settings and confirmation.

(1) ADSTARTE of all scan groups is 0,

(2) TRGMD[0] of scan groups 0,1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$.

(3) SGACT of all scan groups is 0 (before scan groups are started)

### 27.3.14 ADCFnSFTCR — Safety Control Register

ADCFnSFTCR is an 8-bit readable/writable register for safety control.

**Access:** This register can be read/written in 8-bit units.

**Address:** <ADCFn_base> + $3C0_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | RDCLRE | ULEIE | OWEIE | PEIE | IDEIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 27.21 ADCFnSFTCR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | RDCLRE | Read and Clear Enable<br>0: ADCFnDRj and ADCFnDIRj are not cleared by reading ADCFnDRj or ADCFnDIRj.<br>1: ADCFnDRj and ADCFnDIRj are cleared by reading ADCFnDRj or ADCFnDIRj.<br>**CAUTION**<br>ADCFnDIRj.WFLG is cleared by reading ADCFnDRj or ADCFnDIRj regardless of the RDCLRE setting. |
| 3 | ULEIE | Upper-Limit/Lower-Limit Error Interrupt Enable<br>0: Disabled<br>1: Enabled |
| 2 | OWEIE | Overwrite Error Interrupt Enable<br>0: Disabled<br>1: Enabled |
| 1 | PEIE | Parity Error Interrupt Enable<br>0: Disabled<br>1: Enabled |
| 0 | IDEIE | ID Error Interrupt Enable<br>0: Disabled<br>1: Enabled |

**CAUTION**

To prevent malfunction, perform settings of ADCFnSFTCR after the following settings and confirmation.
(1) ADSTARTE of all scan groups is 0,
(2) TRGMD[0] of scan groups 0,1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$.
(3) SGACT of all scan groups is 0 (before scan groups are started)

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.15 ADCFnTDCR —Pin Level Diagnostic Control Register

ADCFnTDCR is an 8-bit readable/writable register that controls the pin level diagnosis.

**Access:** This register can be read/written in 8-bit units.

**Address:** <ADCFn_base> + $3C4_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TDE | — | — | — | — | — | TDLV[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R | R | R | R | R | R/W | R/W |

**Table 27.22 ADCFnTDCR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | TDE | Pin Level diagnosis Enable<br>0: Pin level diagnosis is disabled.<br>1: Pin level diagnosis is enabled.<br>When TDE is set to 1, all analog pins are disconnected from the input buffer.<br>When TDE is set to 0, all analog pins are connected to the input buffer.<br>When TDE is set to 1, the voltage is fixed to the level specified by TDLV[1:0].<br>Performing A/D conversion in this state and checking the A/D converted value allows diagnosis of the path from an analog pin to the ADCF.<br>**CAUTIONS**<br>1. When TDE = 1, be sure to sent both the PUE and PDE bits of ADCFnVCRj to 0<br>2. Set ADCFnTDCR.TDE to 0 before setting the PUE bits of ADCFnVCRj to 1. Similarly, when setting ADCFnTDCR.TDE to 1, set the PUE and PDE bites of ADCFnVCRj for A/D conversion to 0 while ADCFnTDCR.TDE = 1 (the recommended setting is 0 for the PUE and PDE bits of all ADCFnVCRj registers). |
| 6 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | TDLV[1:0] | Pin Level diagnosis Level<br>$0_H$: Even numbers of physical channel groups are discharged to AnVSS, and odd numbers of physical channel groups are charged to AnVCC.<br>$1_H$: Even numbers of physical channel groups are charged to AnVCC, and odd numbers of physical channel groups are discharged to AnVSS.<br>$2_H$: Even numbers of physical channel groups are discharged to AnVSS, and odd numbers of physical channel groups are charged to 1/2*AnVCC.<br>$3_H$: Even numbers of physical channel groups are charged to 1/2*AnVCC, and odd numbers of physical channel groups are discharged to AnVSS. |

**CAUTION**

To prevent malfunction, perform settings of ADCFnTDCR after the following settings and confirmation.

(1) ADSTARTE of all scan groups is 0

(2) TRGMD[0] of scan groups 0,1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$.

(3) SGACT of all scan groups is 0 (before scan groups are started)

## 27.3.16 ADCFnULLMTBR0 to 2 — Upper-Limit/Lower-Limit Table Registers 0 to 2

ADCFnULLMTBR0-2 are 32-bit readable/writable registers that set the upper-limit and lower-limit values of an A/D converted value. Specify any of ADCFnULLMTBR0 to 2 by ULS[1:0] in ADCFnULLMSRx.

**Access:** This register can be read/written in 32-bit units.

**Address:** ADCFnULLMTBR0: <ADCFn_base> + $3CC_H$
ADCFnULLMTBR1: <ADCFn_base> + $3D0_H$
ADCFnULLMTBR2: <ADCFn_base> + $3D4_H$

**Value after reset:** $7FFE\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ULMTB[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LLMTB[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R |

**Table 27.23 ADCFnULLMTBR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | ULMTB[15:0] | Upper Limit Table<br>These bits specify the upper-limit value of an A/D converted value. ULE (upper limit/lower limit error) is set when the following condition is met.<br>ULMTB[15:0] < A/D converted value<br>The ULMTB[15:0] format is the signed fixed-point format regardless of the format of ADCFnDRj. If the signed integer format is selected for the ADCFnDRj format, the ADCFnDRj format is replaced with the signed fixed-point format, and then the values are compared. Note that ULMTB[15] and ULMTB[0] are always fixed to 0. |
| 15 to 0 | LLMTB[15:0] | Lower Limit Table<br>These bits specify the lower-limit value of an A/D converted value. ULE (upper limit/lower limit error) is set when the following condition is met.<br>LLMTB[15:0] > A/D converted value<br>The LLMTB[15:0] format is the signed fixed-point format regardless of the format of ADCFnDRj. If the signed integer format is selected for the ADCFnDRj format, the ADCFnDRj format is replaced with the signed fixed-point format, and then the values are compared. Note that LLMTB[15] and LLMTB[0] are always fixed to 0. |

**CAUTION**

To prevent malfunction, perform settings of ADCFnULLMTBR0 to 2 after the following settings and confirmation.

(1) ADSTARTE of all scan groups is 0,

(2) TRGMD[0] of scan groups 0,1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$.

(3) SGACT of all scan groups is 0 (before scan groups are started)

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

### 27.3.17 ADCFnECR — Error Clear Register

ADCFnECR is an 8-bit write-only register that controls error clear. The register bits are always read as 0.

**Access:** This register can be read in 8-bit units.

**Address:** <ADCFn_base> + $3D8_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ULEC | OWEC | PEC | IDEC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | W | W | W | W |

**Table 27.24 ADCFnECR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | When writing, write the value after reset. |
| 3 | ULEC | Upper Limit/Lower Limit Error Clear<br>Writing 0: Not cleared<br>Writing 1: Cleared |
| 2 | OWEC | Overwrite Error Clear<br>Writing 0: Not cleared<br>Writing 1: Cleared |
| 1 | PEC | Parity Error Clear<br>Writing 0: Not cleared<br>Writing 1: Cleared |
| 0 | IDEC | ID Error Clear<br>Writing 0: Not cleared<br>Writing 1: Cleared |

## 27.3.18 ADCFnULER — Upper-Limit/Lower-Limit Error Register

ADCFnULER is an 8-bit read-only register that indicates upper limit/lower limit errors.

**Access:** This register can be read in 8-bit units.

**Address:** <ADCFn_base> + $3DC_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ULE | — | ULECAP[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.25 ADCFnULER register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | ULE | Upper Limit/Lower Limit Error<br>0: No error<br>1: An error is present.<br>Setting condition<br>The A/D converted value exceeds the range of the specified upper limit/lower limit table<br>Clearing condition<br>A value of 1 is written to ULEC. |
| 6 | Reserved | When read, the value after reset is read. |
| 5 to 0 | ULECAP[5:0] | Upper Limit/Lower Limit Error Capture<br>The virtual channel at the time when an upper limit/lower limit error occurred is captured.<br>Capturing condition<br>ULE = 0 and the A/D converted value exceeds the range of the specified upper limit/lower limit table.<br>Clearing condition<br>A value of 1 is written to ULEC |

**CAUTION**

ADCFnULER is updated when the AD converted value is written to ADCFnDRj.

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.19 ADCFnOWER — Overwrite Error Register

ADCFnOWER is an 8-bit read-only register that indicates an overwrite error.

**Access:** This register can be read in 8-bit units.

**Address:** <ADCFn_base> + $3E0_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | OWE | — | OWECAP[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.26 ADCFnOWER register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | OWE | Overwrite Error<br>0: No error<br>1: An error is present.<br>Setting condition<br>The A/D converted value is written to ADCFnDRj when WFLG = 1<br>Clearing condition<br>A value of 1 is written to OWEC |
| 6 | Reserved | When read, the value after reset is read. |
| 5 to 0 | OWECAP[5:0] | Overwrite Error Capture<br>The virtual channel at the time when an overwrite error occurred is captured.<br>Capturing condition<br>The A/D converted value is written to ADCFnDRj when OWE = 0 and WFLG = 1.<br>Clearing condition<br>A value of 1 is written to OWEC |

**CAUTION**

ADCFnOWER is updated when the AD converted value is written to ADCFnDRj.

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.20 ADCFnPER — Parity Error Register

ADCFnPER is an 8-bit read-only register that indicates a parity error.

**Access:** This register can be read in 8-bit units.

**Address:** <ADCFn_base> + $3E4_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PE | — | PECAP[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.27 ADCFnPER register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | PE | Parity Error<br>0: No error<br>1: An error is present.<br>Setting condition<br>A parity error is detected.<br>Clearing condition<br>A value of 1 is written to PEC. |
| 6 | Reserved | When read, the value after reset is read. |
| 5 to 0 | PECAP[5:0] | Parity Error Capture<br>The virtual channel at the time when a parity error occurred is captured.<br>Capturing condition<br>A parity error is detected when PE = 0.<br>Clearing condition<br>A value of 1 is written to PEC. |

**CAUTION**

ADCFnPER is updated when ADCFnDRj or ADCFnDIRj is read.

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.21 ADCFnIDER — ID Error Register

ADCFnIDER is an 8-bit read-only register that indicates an ID error.

**Access:** This register can be read in 8-bit units.

**Address:** <ADCFn_base> + $3E8_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IDE | — | IDECAP[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.28 ADCFnIDER register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | IDE | ID Error<br>0: No error<br>1: An error is present.<br>Setting condition<br>The physical channel specified in ADCFnVCRj does not match the physical channel actually converted.<br>Clearing condition<br>A value of 1 is written to IDEC. |
| 6 | Reserved | When read, the value after reset is read. |
| 5 to 0 | IDECAP[5:0] | ID Error Capture<br>The virtual channel at the time when an ID error occurred is captured.<br>Capturing condition<br>The physical channel specified in ADCFnVCRj does not match the physical channel actually converted when IDE = 0.<br>Clearing condition<br>A value of 1 is written to IDEC. |

**CAUTION**

ADCFnIDER is updated when the AD converted value is written to ADCFnDRj.

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.22 ADCFnSGSTCRx — Scan Group x Start Control Register

ADCFnSGSTCRx is an 8-bit write-only register that controls the start of scan group x. The register bits are always read as 0.

**Access:** This register can be written in 8-bit units.

**Address:** ADCFnSGSTCR0: <ADCFn_base> + $480_H$
ADCFnSGSTCR1: <ADCFn_base> + $500_H$
ADCFnSGSTCR2: <ADCFn_base> + $580_H$
ADCFnSGSTCR3: <ADCFn_base> + $600_H$
ADCFnSGSTCR4: <ADCFn_base> + $680_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SGST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 27.29 ADCFnSGSTCRx register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SGST | Scan Group Start<br>Condition for starting scan group x:<br>A value of 1 is written to SGST when SGACT = 0 |

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

### 27.3.23 ADCFnADTSTCRy — AD Timer y Start Control Register

ADCFnADTSTCRy is an 8-bit write-only register that controls the start of AD timer y. The register bits are always read as 0.

**Access:** This register can be written in 8-bit units.

**Address:** ADCFnADTSTCR3: <ADCFn_base> + $608_H$
ADCFnADTSTCR4: <ADCFn_base> + $688_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ADTST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 27.30 ADCFnADTSTCRy register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | ADTST | A/D Timer Start<br>Condition for starting A/D timer y:<br>A value of 1 is written to ADTST when ADTACT = 0 |

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.24 ADCFnADTENDCRy — AD Timer y End Control Register

ADCFnADTENDCRy is an 8-bit write-only register that controls the end of the AD timer y. The register bits are always read as 0.

**Access:** This register can be written in 8-bit units.

**Address:** ADCFnADTENDCR3: <ADCFn_base> + $60C_H$
ADCFnADTENDCR4: <ADCFn_base> + $68C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ADTEND |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | W |

**Table 27.31 ADCFnADTENDCRy register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | When writing, write the value after reset. |
| 0 | ADTEND | A/D Timer End<br>Condition for finishing A/D timer y:<br>A value of 1 is written to ADTEND when ADTACT = 1 |

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.25 ADCFnSGCRx — Scan Group x Control Register

ADCFnSGCRx is an 8-bit readable/writable register that controls scan group x.

**Access:** This register can be read/written in 8-bit units.

**Address:** ADCFnSGCR0: <ADCFn_base> + $490_H$
ADCFnSGCR1: <ADCFn_base> + $510_H$
ADCFnSGCR2: <ADCFn_base> + $590_H$
ADCFnSGCR3: <ADCFn_base> + $610_H$
ADCFnSGCR4: <ADCFn_base> + $690_H$

**Value after reset:** $00_H$

- When x = 0 to 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | ADSTARTE | SCANMD | ADIE | — | — | — | TRGMD[0] |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R | R | R | R/W |

**Table 27.32 ADCFnSGCRx register contents (x = 0 to 2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 | ADSTARTE | Scan Group Synchronization Start Enable<br>0: ADSTART is disabled.<br>1: ADSTART is enabled. |
| 5 | SCANMD | Scan Mode<br>0: Multicycle scan mode<br>1: Continuous scan mode<br>In multicycle scan mode, scans are repeated as many times as specified in ADCFnSGMCYCRx. In continuous scan mode, scans are repeated with no limit of times. |
| 4 | ADIE | Scan End Interrupt Enable<br>0: ADInx is not output at the end of scan for SGx.<br>1: ADInx is output at the end of scan for SGx.<br>ADIE of ADCFnSGCRx is independent of ADIE of ADCFnVCRj. For details, see **Section 27.4.15, Scan End Interrupt Request** |
| 3 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | TRGMD[0] | Trigger Mode<br>$0_H$: Trigger input to SGx is disabled.<br>$1_H$: The SGx_TRG hardware trigger is selected for the trigger input to SGx. |

- When x = 3, 4

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADTSTARTE | ADSTARTE | SCANMD | ADIE | — | — | TRGMD[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R | R | R/W | R/W |

**Table 27.33 ADCFnSGCRx register contents (x = 3, 4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | ADTSTARTE | AD Timer Synchronization Start Enable<br>0: ADSTART is disabled.<br>1: ADSTART is enabled. |
| 6 | ADSTARTE | Scan Group Synchronization Start Enable<br>0: ADSTART is disabled.<br>1: ADSTART is enabled. |
| 5 | SCANMD | Scan Mode<br>0: Multicycle scan mode<br>1: Continuous scan mode<br>In multicycle scan mode, scans are repeated as many times as specified in ADCFnSGMCYCRx. In continuous scan mode, scans are repeated with no limit of times. |
| 4 | ADIE | Scan End Interrupt Enable<br>0: ADInx is not output at the end of scan for SGx.<br>1: ADInx is output at the end of scan for SGx.<br>ADIE of ADCFnSGCRx is independent of ADIE of ADCFnVCRj. For details, see **Section 27.4.15, Scan End Interrupt Request**. |
| 3, 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | TRGMD[1:0] | Trigger Mode<br>$0_H$: Trigger input to SGx is disabled.<br>Trigger input to the AD timer x is disabled.<br>$1_H$: The SGx_TRG hardware trigger is selected for the trigger input to SGx.<br>Trigger input to the AD timer x is disabled.<br>$2_H$: The AD timer trigger x is selected for the trigger input to SGx.<br>Trigger input to AD timer x is disabled.<br>$3_H$: The AD timer trigger x is selected for the trigger input to SGx.<br>The SGx_TRG hardware trigger is selected for the trigger input to AD timer x. |

**CAUTIONS**

1. To prevent malfunctions, make settings for SCANMD and ADIE of ADCDnSGCRx after making or confirming the following settings.
   (1) ADSTARTE of scan group x is 0 and TRGMD of scan group x is $0_H$.
   (2) SGACT of scan group x is 0 (before the scan group is started).
2. If a trigger of lower-priority scan group is input to a scan group for which continuous scan mode is set (SCANMD = $1_H$), the trigger is not accepted. Therefore, it is assumed that continuous scan mode is set to scan group 0.

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.26 ADCFnSGVCSPx — Scan Group x Start Virtual Channel Pointer

ADCFnSGVCSPx is an 8-bit readable/writable register that specifies the start pointer of a virtual channel.

**Access:** This register can be read/written in 8-bit units.

**Address:** ADCFnSGVCSP0: <ADCFn_base> + $494_H$
ADCFnSGVCSP1: <ADCFn_base> + $514_H$
ADCFnSGVCSP2: <ADCFn_base> + $594_H$
ADCFnSGVCSP3: <ADCFn_base> + $614_H$
ADCFnSGVCSP4: <ADCFn_base> + $694_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | VCSP[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.34 ADCFnSGVCSPx register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 to 0 | VCSP[5:0] | Start Virtual Channel Pointer<br>These bits select the virtual channel from which the scan is to be started. When SGx is started, processing for the virtual channels from ADCFnSGVCSPx to ADCFnSGVCEPx is executed. |

**CAUTIONS**

1. ADCFnSGVCSPx must be equal to or smaller than ADCFnSGVCEPx.
2. To prevent malfunction, perform settings of ADCFnSGVCSPx after the following settings and confirmation.
   (1) ADSTARTE of all scan groups is 0
   (2) TRGMD[0] of scan groups 0,1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$.
   (3) SGACT of all scan groups is 0 (before scan groups are started)
3. Do not set a value greater than the number of virtual channels provided.

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.27 ADCFnSGVCEPx — Scan Group x End Virtual Channel Pointer

ADCFnSGVCEPx is an 8-bit readable/writable register that specifies the end pointer of a virtual channel.

**Access:** This register can be read/written in 8-bit units.

**Address:** ADCFnSGVCEP0: <ADCFn_base> + $498_H$
ADCFnSGVCEP1: <ADCFn_base> + $518_H$
ADCFnSGVCEP2: <ADCFn_base> + $598_H$
ADCFnSGVCEP3: <ADCFn_base> + $618_H$
ADCFnSGVCEP4: <ADCFn_base> + $698_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | VCEP[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.35 ADCFnSGVCEPx register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 to 0 | VCEP[5:0] | End Virtual Channel Pointer<br>These bits select the virtual channel from which the scan is to be ended.<br>When SGx is started, processing for the virtual channels from ADCFnSGVCSPx to ADCFnSGVCEPx is executed. |

**CAUTIONS**

1. ADCFnSGVCSPx must be equal to or smaller than ADCFnSGVCEPx.
2. To prevent malfunction, perform settings of ADCFnSGVCEPx after the following settings and confirmation.
   (1) ADSTARTE of all scan groups is 0
   (2) TRGMD[0] of scan groups 0,1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$.
   (3) SGACT of all scan groups is 0 (before scan groups are started)
3. Do not set a value greater than the number of virtual channels provided.

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.28 ADCFnSGMCYCRx — Scan Group x Multicycle Register

ADCFnSGMCYCRx is an 8-bit readable/writable register that specifies the number of scan times in multicycle scan mode.

**Access:** This register can be read/written in 8-bit units.

**Address:** ADCFnSGMCYCR0: <ADCFn_base> + $49C_H$
ADCFnSGMCYCR1: <ADCFn_base> + $51C_H$
ADCFnSGMCYCR2: <ADCFn_base> + $59C_H$
ADCFnSGMCYCR3: <ADCFn_base> + $61C_H$
ADCFnSGMCYCR4: <ADCFn_base> + $69C_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | MCYC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.36 ADCFnSGMCYCRx register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 0 | MCYC[7:0] | Multicycle<br>These bits specify the number of scan times in multicycle scan mode<br>Number of scan times = MCYC[7:0] + 1<br>When SGx is started, scans are repeated for virtual channels from ADCFnSGVCSPx to ADCFnSGVCEPx as many times as specified in ADCFnSGMCYCRx. |

**CAUTION**

To prevent malfunction, perform settings of ADCFnSGMCYCRx after the following settings and confirmation.

(1) ADSTARTE of all scan groups is 0

(2) TRGMD[0] of scan groups 0,1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$.

(3) SGACT of all scan groups is 0 (before scan groups are started)

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.29 ADCFnSGSRx — Scan Group x Status Register

ADCFnSGSRx is an 8-bit read-only register that indicates the status of scan group x.

**Access:** This register can be read in 8-bit units.

**Address:** ADCFnSGSR0: <ADCFn_base> + $4A4_H$
ADCFnSGSR1: <ADCFn_base> + $524_H$
ADCFnSGSR2: <ADCFn_base> + $5A4_H$
ADCFnSGSR3: <ADCFn_base> + $624_H$
ADCFnSGSR4: <ADCFn_base> + $6A4_H$

**Value after reset:** $00_H$

- x = 0 to 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | SGACT | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.37 ADCFnSGSRx register contents (x = 0 to 2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is read. |
| 1 | SGACT | Scan Group Status<br>0: There is no conversion in SGx pending.<br>1: There is a conversion in SGx pending.<br>If this bit is 1, the status of SGx is either of the following:<br>• trigger request is held pending<br>• the conversion is ongoing<br>• the conversion is held pending due to occurrence a higher priority conversion |
| 0 | Reserved | When read, the value after reset is read. |

- x = 3, 4

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | ADTACT | SGACT | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 27.38 ADCFnSGSRx register contents (x = 3, 4)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 3 | Reserved | When read, the value after reset is read. |
| 2 | ADTACT | A/D Timer Status<br>0: A/D timer x is in idle state.<br>1: A/D timer x is running |
| 1 | SGACT | Scan Group Status<br>0: There is no conversion in SGx pending.<br>1: There is a conversion in SGx pending.<br>If this bit is 1, the status of SGx is either of the following:<br>• trigger request is held pending<br>• the conversion is ongoing<br>• the conversion is held pending due to occurrence a higher priority conversion |
| 0 | Reserved | When read, the value after reset is read. |

**CAUTION**

**The SGSRx status flags will be set with a certain delay after the set event has occurred.**

**For the delay time refer to Section 27.4.9, Analog Input Sampling and Scan Group Processing Time.**

**Consider this behavior when reading the status of this bit.**

**Alternatively use the corresponding interrupt / interrupt status flag to monitor the conversion status.**

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.30 ADCFnADTIPRy — AD Timer Initial Phase Register y

ADCFnADTIPRy is a 32-bit readable/writable register that sets the initial phase of A/D timer y.

**Access:** This register can be read/written in 32-bit units.

**Address:** ADCFnADTIPR3: <ADCFn_base> + $628_H$
ADCFnADTIPR4: <ADCFn_base> + $6A8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | ADTIP[20:16] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADTIP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.39 ADCFnADTIPRy register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 20 to 0 | ADTIP[20:0] | A/D Timer Initial Phase<br>These bits set the initial phase of A/D timer y.<br>(1) After A/D timer y is started, ADCFnADTIPRy is loaded to A/D timer y and the timer counts down.<br>(2) After A/D timer y becomes 0, A/D timer trigger y is output for one cycle, ADCFnADTPRRy is loaded to A/D timer y, and the timer counts down again.<br>After that, (2) is repeated.<br>For details, see **Section 27.4.6, Example of A/D Timer Operation**. |

**CAUTION**

To prevent malfunction, perform ADCFnADTIPRy settings when ADTACT of scan group y is 0 (before A/D timer is started), ADTSTARTE of scan group y is 0, and TRGMD[1:0] of scan group y is not $3_H$.

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.31 ADCFnADTPRRy — AD Timer Period Register y

ADCFnADTPRRy is a 32-bit readable/writable register that sets the cycle of A/D timer y.

**Access:** This register can be read/written in 32-bit units.

**Address:** ADCFnADTPRR3: <ADCFn_base> + $62C_H$
ADCFnADTPRR4: <ADCFn_base> + $6AC_H$

**Value after reset:** $001F\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | ADTPR[20:16] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADTPR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 27.40 ADCFnADTPRRy register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 20 to 0 | ADTPR[20:0] | A/D Timer Cycle<br>These bits set the cycle of A/D timer y.<br>(1) After A/D timer y is started, ADCFnADTIPRy is loaded to A/D timer y and the timer counts down.<br>(2) After A/D timer y becomes 0, A/D timer trigger y is output for one cycle, ADCFnADTPRRy is loaded to A/D timer y, and the timer counts down again.<br>After that, (2) is repeated.<br>For details, see **Section 27.4.6, Example of A/D Timer Operation**. |

**CAUTION**

To prevent malfunction, perform ADTPPRy settings when ADTACT of scan group y is 0 (before A/D timer is started), ADTSTARTE of scan group y is 0, and TRGMD[1:0] of scan group y is not $3_H$.

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.3.32 ADCFnULLMSRx — Scan Group x Upper-Limit/Lower-Limit Table Select Register

ADCFnULLMSRx is an 8-bit readable/writable register that controls scan group x.

**Access:** This register can be read/written in 8-bit units.

**Address:** ADCFnULLMSR0: <ADCFn_base> + $4B0_H$
ADCFnULLMSR1: <ADCFn_base> + $530_H$
ADCFnULLMSR2: <ADCFn_base> + $5B0_H$
ADCFnULLMSR3: <ADCFn_base> + $630_H$
ADCFnULLMSR4: <ADCFn_base> + $6B0_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | ULS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R/W | R/W |

**Table 27.41 ADCFnULLMSRx register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | ULS | Upper Limit/Lower Limit Table Select<br>$0_H$: Neither upper limit nor lower limit is checked.<br>$1_H$: Upper limit and lower limit are checked in ADCFnULLMTBR0.<br>$2_H$: Upper limit and lower limit are checked in ADCFnULLMTBR1.<br>$3_H$: Upper limit and lower limit are checked in ADCFnULLMTBR2.<br>Upper limit and lower limit are checked by using the upper limit/lower limit table selected by ULS[1:0] when storing the A/D converted value in ADCFnDRj. |

**CAUTION**

To prevent malfunction, perform settings of ADCFnULLMSRx after the following settings and confirmation.

(1) ADSTARTE of all scan groups is 0

(2)TRGMD[0] of scan groups 0, 1, 2 is $0_H$ and TRGMD[1:0] of scan groups 3, 4 is $0_H$.

(3) SGACT of all scan groups is 0 (before scan groups are started)

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

# 27.4 Function

## 27.4.1 Examples of Normal AD Conversion Operation

### 27.4.1.1 Multicycle Scan Mode

The following figure shows an example of operation when converting four virtual channels by 2-cycle scan for scan group 0 in multicycle scan mode by using normal A/D conversion mode (CNVCLS[2:0] = $0_H$).

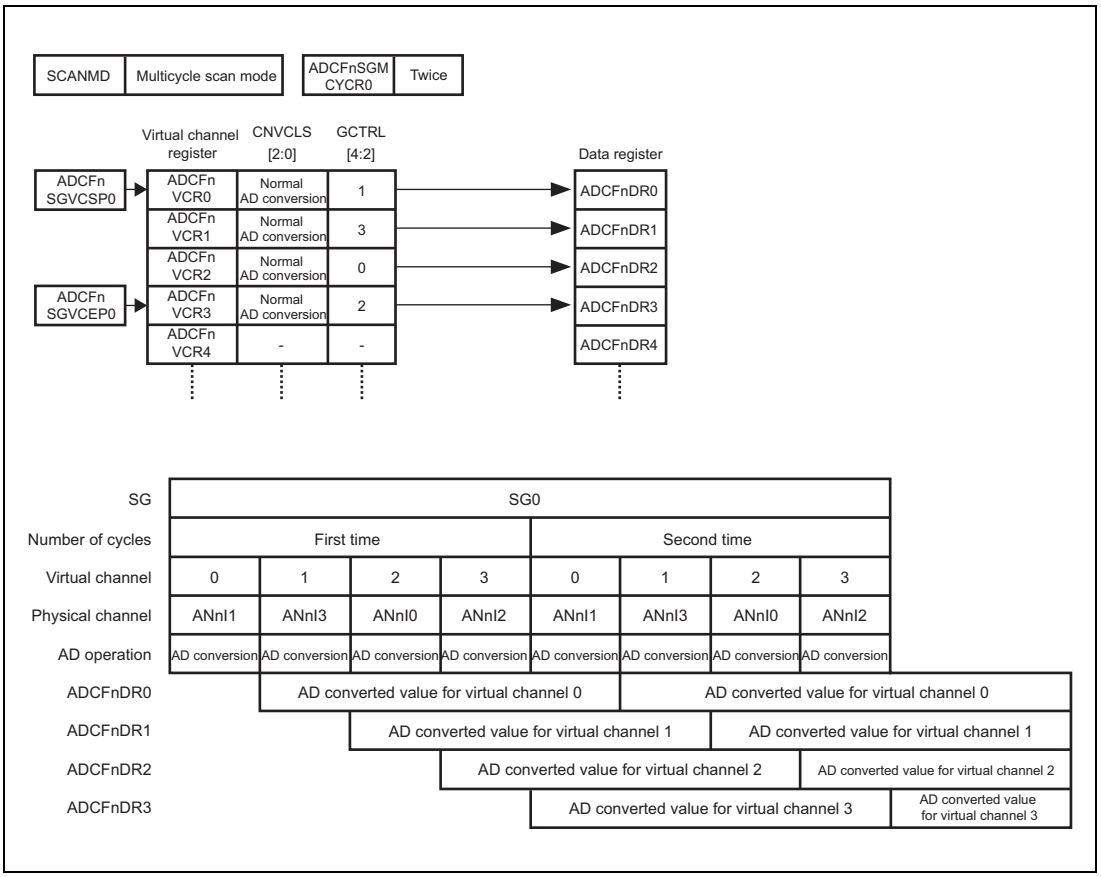



**Figure 27.4 Example of Operation in Multicycle Scan Mode**

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

### 27.4.1.2 Continuous Scan Mode

**Figure 27.5** shows an example of operation when converting four virtual channels for scan group 0 in continuous scan mode by using normal A/D conversion mode (CNVCLS[2:0] = $0_H$).

In this mode, if a trigger of a lower-priority scan group is input to a scan group for which continuous scan mode is set (SCANMD = $1_H$), the trigger is not accepted. Therefore, it is assumed that continuous scan mode is set for scan group 0.

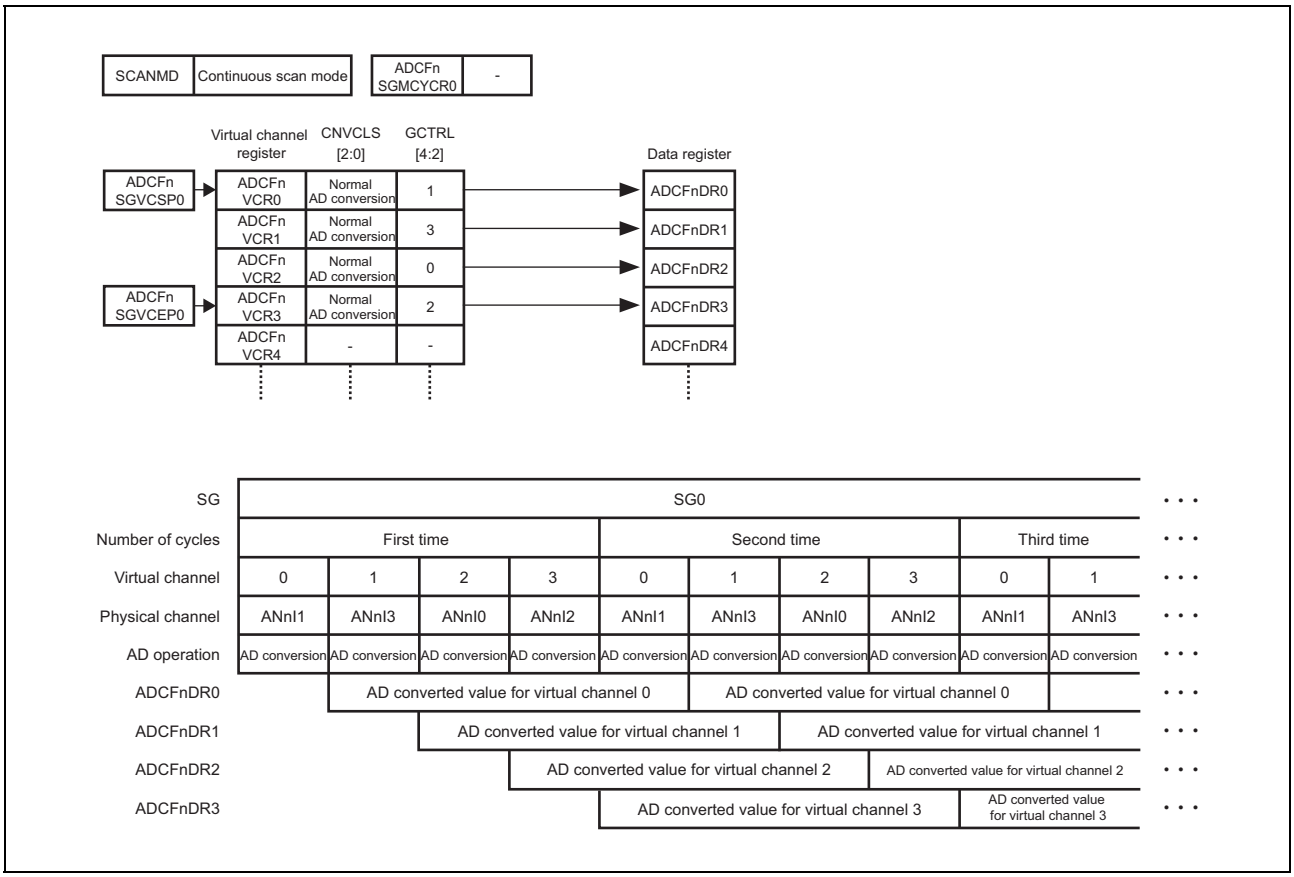



**Figure 27.5 Example of Operation in Continuous Scan Mode**

### 27.4.2 Example of Normal A/D Conversion Operation in Addition Mode

**Figure 27.6** shows an example of operation when converting four virtual channels for scan group 0 by using normal A/D conversion (CLVCLS[2:0] = $4_H$) in addition mode.

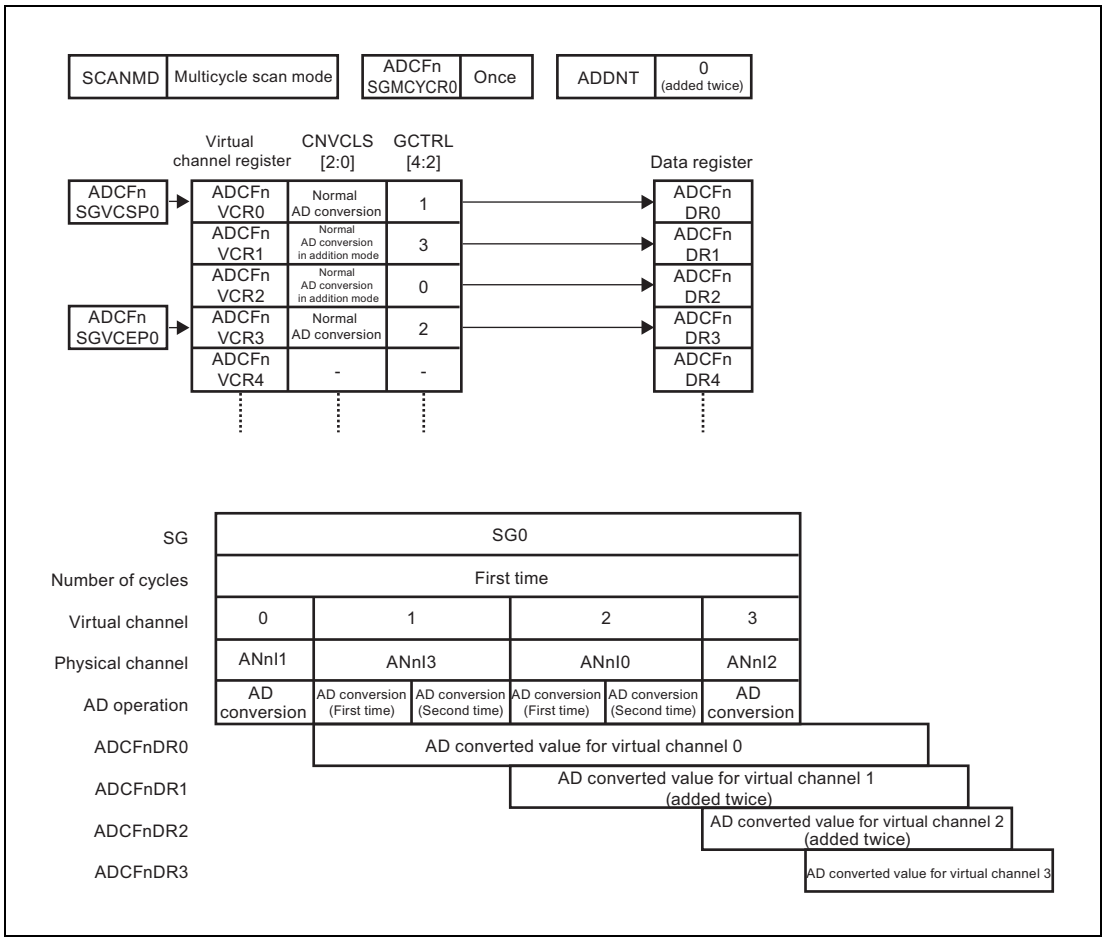


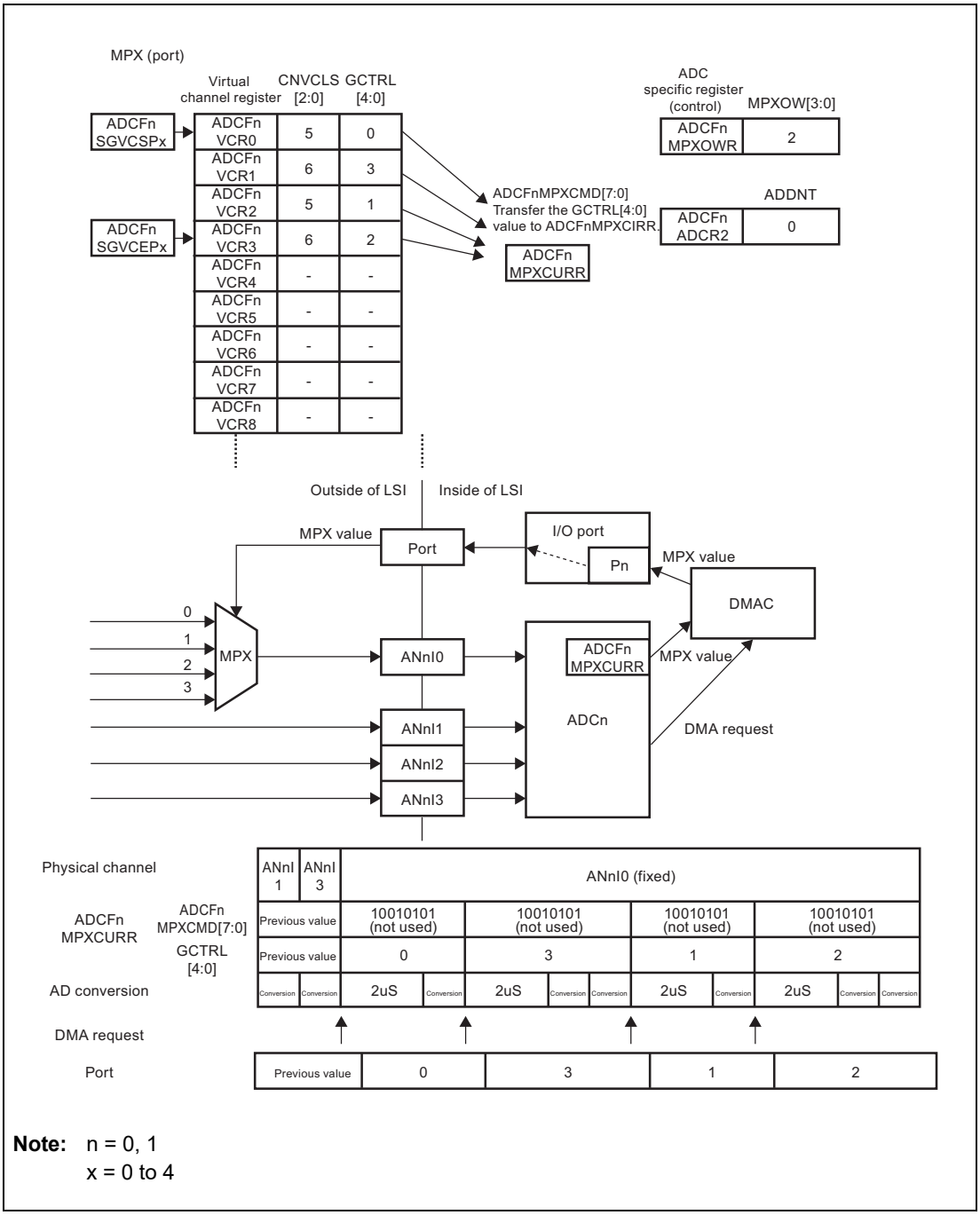

**Figure 27.6 Example of Normal A/D Conversion Operation in Addition Mode**

### 27.4.3 Example of Operation of External Analog Multiplexer

The following shows examples of operation of an external analog multiplexer using normal A/D conversion with the MPX mode (CNVCLS[2:0] = $5_H$) or normal A/D conversion with the MPX mode of addition mode (CNVCLS[2:0] = $6_H$).

#### 27.4.3.1 Example of Using an External Analog Multiplexer (Port Output)

**Figure 27.7** shows an example of port output using an external analog multiplexer.




**Note:** n = 0, 1
x = 0 to 4

**Figure 27.7 Example of Using an External Analog Multiplexer (Port Output)**

The input channel for the external analog multiplexer (MPX) is fixed to one dedicated physical input channel and is different for the implemented ADCFn modules:

- The physical input channel for MPX for ADCF0 is AN0I06
- The physical input channel for MPX for ADCF1 is AN1I00

#### 27.4.3.2 Usage Notes for an External Analog Multiplexer

When you use an external MPX, conform to the following notes so as not to cause system crash.

Except for the exceptions below, set the MPX wait as follows:

- An MPX value is transferred at a port: Insert a wait of 1usec or more.

Exception 1) When an external MPX is used for one scan group

If an external MPX is used for SG0 when SUSMTD[1:0] = $1_H$, or if an external MPX is used for any of SG from SG0 to SG3 when SUSMTD[1:0] = $2_H$, set the MPX wait as follows:

- An MPX value is transferred at a port: Insert a wait of 1usec or more.

Exception 2) When an external MPX is used for multiple scan groups

If an external MPX is used when SUSMTD[1:0] = $1_H$ or $2_H$, conform to the following notes:

- For the start virtual channel of each scan group, the external MPX should be disabled. (However, for the start virtual channel of the scan group whose priority is the lowest among the scan groups for which the external MPX is used, the MPXE bit can be set to 1, which does not cause problems.)

Furthermore, set the MPX wait as follows.

- An MPX value is transferred at a port: Insert a wait of 1usec or more.

### 27.4.4 Example of Synchronous Suspend and Resume Operation

**Figure 27.8** shows an example of synchronous suspend and resume operation when a higher-priority SG interrupts a lower-priority SG.

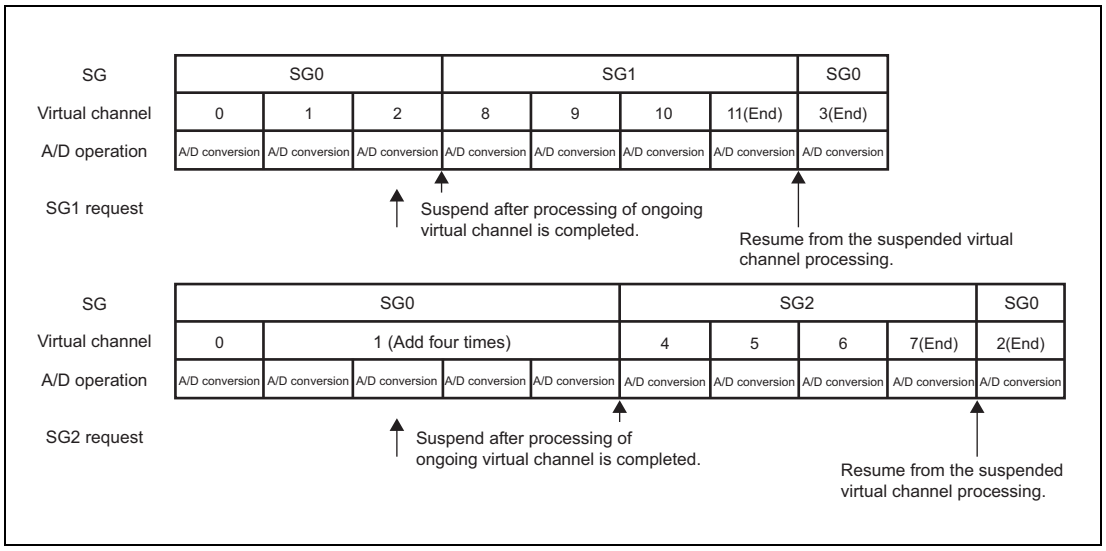



**Figure 27.8 Example of Synchronous Suspend and Resume Operation**

**CAUTION**

**Priority of scan groups:**

**Low High**

**SG0 < SG1 < SG2 < SG3 < SG4**

### 27.4.5 Example of Asynchronous Suspend and Resume Operation

**Figure 27.9** shows an example of asynchronous suspend and resume operation when a higher-priority SG interrupts a lower-priority SG.

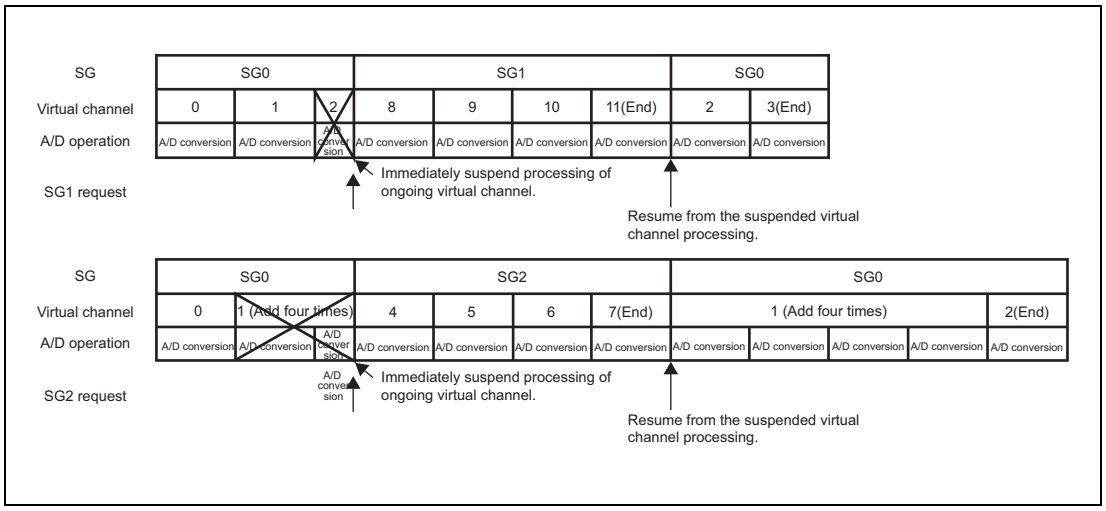



**Figure 27.9 Example of Asynchronous Suspend and Resume Operation**

**CAUTION**

**Priority of scan groups:**

**Low High**

**SG0 < SG1 < SG2 < SG3 < SG4**

### 27.4.6 Example of A/D Timer Operation

**Figure 27.10** shows an example of A/D timer operation.

The A/D timer counts are based on the internal ADC clock ADCFCLK.

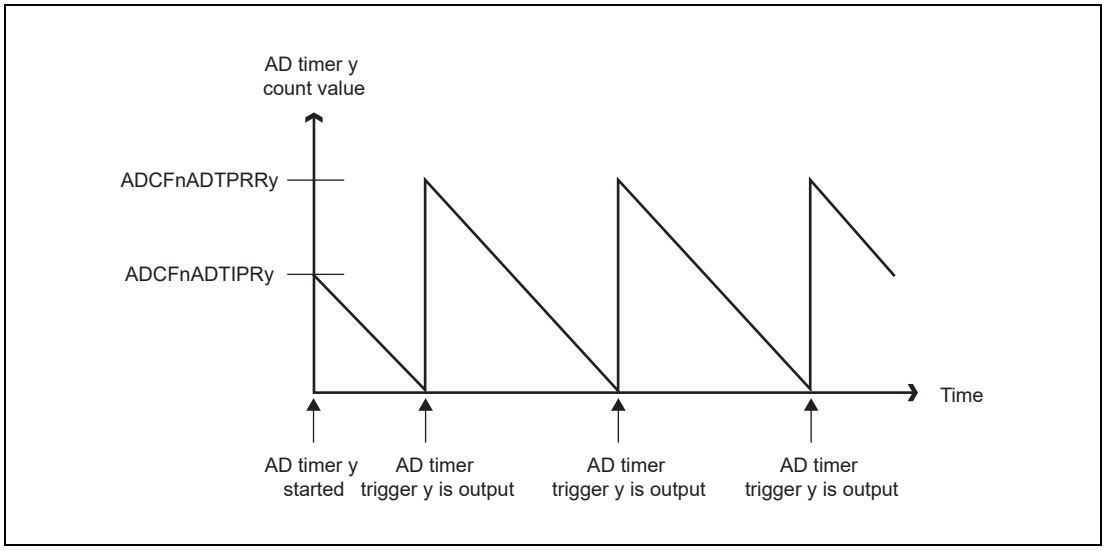



**Figure 27.10 Example of A/D Timer Operation**

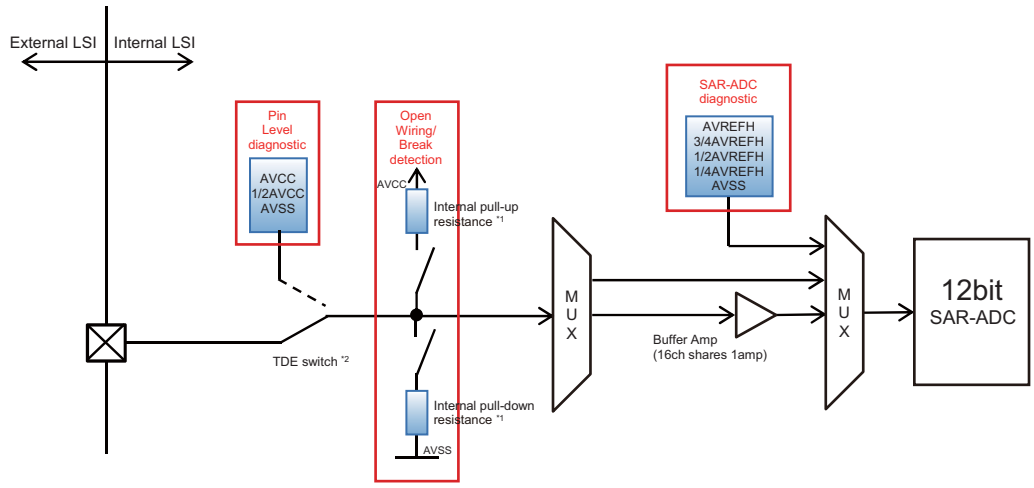
**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

## 27.4.7 Diagnostic Function

The ADCF is equipped with the following diagnostic functions.

- PIN level diagnostic function
- SAR-ADC diagnostic function
- Open wiring and break detection




Note 1. For Internal pull-up/down resistance data, see **Section 35.4.3, Analog Input Pins (ADCnIm) Characteristics**.

Note 2. For TDE switch resistance data, see **Section 35.7.3, A/D Converter Characteristics**.

**Figure 27.11 diagnosis circuits**

### 27.4.7.1 Pin Level Diagnostic Function

The pin level diagnostic function performs A/D conversion that is set to a different voltage for even-number physical channel groups and odd-number physical channel groups to check an abnormal path from the ANI.

The different voltage setting is made in ADCFnTDCR and can be detected by combination of AnVSS, AnVCC, and 1/2 × AnVCC.

Features and settings of the pin level diagnostic function of the ADCF are described below.

**[Features]**

1. Users can select desired physical channels to be tested.
2. AnVSS, AnVCC, or 1/2AnVCC is selectable as a diagnosis level.
3. Performing A/D conversion for SG0 to SG4 makes the pin level diagnostic function available.

**[Settings]**

1. Make settings according to the initial settings (**Figure 27.19, Initial Settings**).
2. Set CNVCLS[2:0] = $0_H$ and optional channels for GCTRL[4:0] in the virtual channel setting register ADCFnVCRj.

3. Set TDE = $1_H$ and optional pin level diagnosis level for TDLV[1:0] in the pin level diagnostic function setting register.
4. Make other settings required for A/D conversion according to the initial settings (**Figure 27.19, Initial Settings**).
5. Assert triggers of SG0 to SG4 and perform A/D conversion.

### 27.4.7.2 A/D Conversion Circuit Diagnostic Function

The A/D conversion circuit diagnostic function is used to verify that A/D conversion operates correctly.

The voltage value setting is made in GCTRL[4:0] when CNVCLS[2:0] = $3_H$, and can be converted for AnVREFH × 1, AnVREFH × 3/4, AnVREFH × 1/2, AnVREFH × 1/4, and AnVSS.

Features and settings of the A/D conversion circuit diagnostic function of the ADCF are described below.

**[Features]**

1. AnVREFH × 1, AnVREFH × 3/4, AnVREFH × 1/2, AnVREFH × 1/4, or AnVSS is selectable as a diagnosis voltage level.
2. Performing A/D conversion for SG0 to SG4 makes the A/D conversion circuit diagnostic function available.

**[Settings]**

1. Make settings according to the initial settings (**Figure 27.19, Initial Settings**)
2. Set CNVCLS[2:0] = $3_H$ and optional diagnosis voltage level for GCTRL[4:0] in the virtual channel setting register ADCFnVCRj (j = 0 to 35).
3. Make other settings required for A/D conversion according to the initial settings (**Figure 27.19, Initial Settings**).
4. Assert triggers of SG0 to SG4 and perform A/D conversion.

**CAUTION**

**When an injection current is applied on a pin that A/D conversion is operated just before using A/D conversion circuit diagnostic function, the accuracy of A/D conversion in the diagnostic function may be affected. So A/D conversion must be operated for a pin without injection current before using A/D conversion circuit diagnostic function.**

**NOTE**

When the input voltage exceeds the supply voltage or fall below the ground voltage, injection current occurs.

(1) When both A/D conversion and A/D conversion circuit diagnostic is operated in the same SG, a pin of no injection current must be selected before A/D conversion circuit diagnostic.

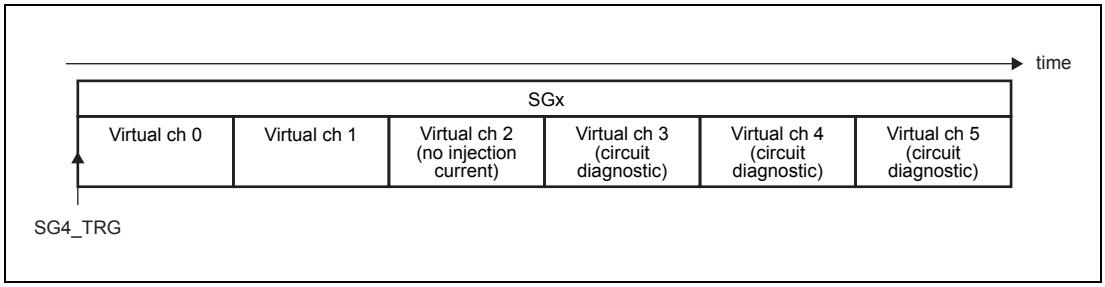



**Figure 27.12 Example of SG with both A/D conversion and A/D conversion circuit diagnostic function in the same SG**

(2) When a SG has higher priority than SG of A/D conversion circuit diagnostic, A/C conversion circuit diagnostic may be suspended and resumed by the operation of SG with higher priority. In this case, execute A/D conversion for a pin*1 without injection current at the last of SG with higher priority.

**Note 1.** Possible to use an unused pin instead of a pin of no injection current

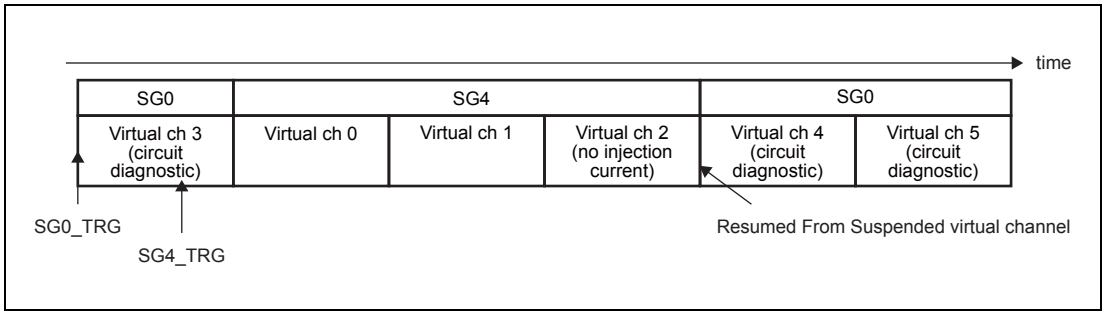



**Figure 27.13 Example of SG with higher priority than SG of A/D conversion circuit diagnostic**

### 27.4.7.3 Wiring-Break Detection Diagnostic Function

The wiring-break detection diagnostic function detects wiring-break of the ANI. The pull-down and pull-up methods are used to detect a wiring break. If a wiring break is present when using the pull-down method, the A/D conversion result is lowered to and kept at approximately 0 V, which is the prescribed minimum level, which indicates that a wiring break has occurred. If a wiring break is present when using the pull-up method, the A/D conversion result is raised to and kept at approximately AnVCC, which is the prescribed maximum level, which indicates that a wiring break has occurred. The pull-up and pull-down is active during the sampling time tSPL. And During the conversion time tSAR the resistors are disconnected.

**[Feature]**

1. Users can select desired physical channels for which wiring-break is to be detected.

**[Settings]**

1. Make settings according to the initial settings (**Figure 27.19, Initial Settings**).
2. Set CNVCLS[2:0] = $0_H$ PDE to $1_H$, PUE to $0_H$, and optional channels for GCTRL[4:0] in the virtual channel setting register ADCFnVCRj (j = 0 to 35)

3. Set CNVCLS[2:0] to $0_H$, PDE to $0_H$, PUE to $1_H$, and specify the physical channels set in step 2 for GCTRL[4:0] in virtual channel setting registers ADCFnVCR(j+1) (j = 0 to 35).
4. Set CNVCLS[2:0] to $0_H$, PDE to $0_H$, PUE to $0_H$, and specify the physical channels set in step 2 for GCTRL[4:0] in virtual channel setting registers ADCFnVCR (j + 2) (j = 0 to 35).
5. Assert the corresponding triggers of SG0 to SG4, and then perform regular A/D conversion based on the settings specified in the ADCFnVCRj to ADCFnVCR (j + 2) registers.
6. The CPU compares the result of A/D conversion performed based on ADCFnVCRj (pull-down method) and the result of A/D conversion performed based on ADCFnVCR (j + 1) (pull-up method), compares the result of A/D conversion performed based on ADCFnVCR (j + 1) (pull-up method) and the result of A/D conversion performed based on ADCFnVCR (j + 2) (regular A/D conversion result), and then determines whether or not a wiring break has occurred.

**NOTE**

This function charge/discharge external capacitor. The next conversion need to consider signal settling time.

To prevent misdetection of wire-break of the ANI, the analog input need to be kept constant voltage level during wiring-break detection diagnosis.

### 27.4.8 ADC conversion time

Below table shows the ADC conversion time under different internal clock frequency. For more details see clock chapter.

**Table 27.42 ADC conversion time**

| ADCFCLK | Total conversion time |
|---|---|
| 40 MHz (when CPU clock is 240MHz/160MHz) | 1.00 µs |
| 30 MHz (when CPU clock is 120MHz) | 1.33 µs |

### 27.4.9 Analog Input Sampling and Scan Group Processing Time

ADCF has embedded-sample & hold circuit. After SGST bit of ADCFnSGCRx is set to 1 and scan group start delay time ($t_D$) has passed, the ADC executes sampling and then starts sequential compare transition processing.

Scan group processing time ($t_{SG}$) includes scan group start delay time ($t_D$), sampling time ($t_{SPL}$), sequential compare conversion processing time ($t_{SAR}$), and scan group end delay time ($t_{ED}$). **Table 27.43, Scan Group Processing Time** shows scan group processing time.

In multicycle scan mode, the scan group processing time ($t_{SG}$) can be calculated in the formula below when i = the number of virtual channels and j = the number of multicycles.

$t_{SG} = t_D + (t_{SPL} + t_{SAR}) \times i \times j + t_{ED}$

1st cycle scan in continuous scan mode: $t_D + (t_{SPL} + t_{SAR}) \times i$

2nd and onward scan in continuous scan mode: $(t_{SPL} + t_{SAR}) \times i$

Table 27.43 Scan Group Processing Time

| Item | Symbol | Period | Unit |
|---|---|---|---|
| Scan group start delay time | $t_D$ | (2 to 4) × Pϕ + 5 × Iϕ | Pϕ (CLK_LSB)<br>Iϕ (ADCFCLK) |
| Sampling time | $t_{SPL}$ | 18 × Iϕ | Iϕ (ADCFCLK) |
| Sequential compare conversion processing time | $t_{SAR}$ | 22 × Iϕ | Iϕ (ADCFCLK) |
| Scan group end delay time | $t_{ED}$ | (2 to 4) × Iϕ + 3 × Pϕ | Pϕ (CLK_LSB)<br>Iϕ (ADCFCLK) |
| Scan group processing time | $t_{SG}$ | 47 × Iϕ + 5 × Pϕ to 49 × Iϕ + 7 × Pϕ | Pϕ (CLK_LSB)<br>Iϕ (ADCFCLK) |

### 27.4.10 Hardware Trigger Functions

For each scan group x a trigger input SGx_TRG is provide, that starts the conversion of the respective scan group's channels.

The scan group conversion trigger signals SGx_TRG can be generated by several sources:

- Scan groups SG0 to SG4
  External trigger signals ADTRGn.
  For details about the connected trigger signals see **Section 26, Peripheral Interconnect (PIC)**.
- Scan groups SG3 to SG4
  SG3, SG4 can be started by A/D timer
  AD time can be started by the trigger signals from PIC

The external trigger signals ADCFnTRGx are passed through digital noise filters to eliminate noise and signal glitches.

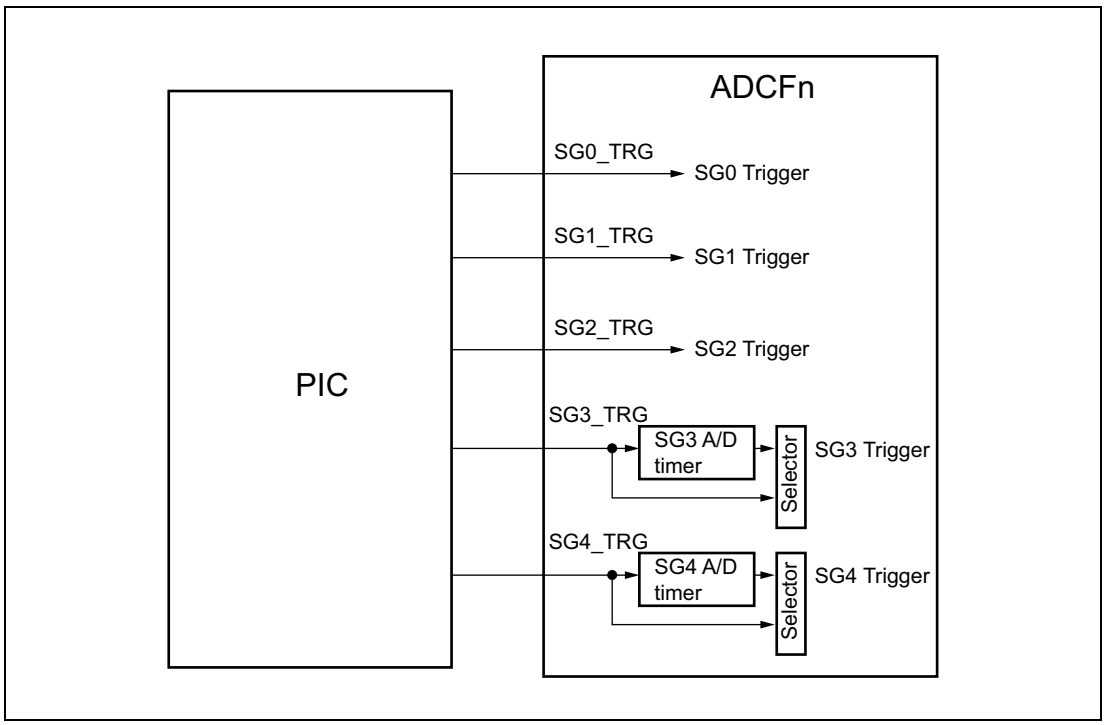



Figure 27.14 Outline of A/D Converter Hardware Trigger Selection

### 27.4.11 Starting a Scan Group by Using a Hardware Trigger

Scan group x can be started by using an input from hardware trigger SGx_TRG. To start scan group x by an input from hardware trigger SGx_TRG, set TRGMD in ADCFnSGCRx to $1_H$. When the selected hardware trigger SGx_TRG is input in this state, SGACT is set to 1. The timing since SGACT is set to 1 until scan group x is started is the same as the timing when SGST is set to 1 by a software trigger.

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.
When an SGx start trigger is generated during scanning (ADCFnSGSRx.SGACT = 1) for same SGx, the SGx start trigger is ignored.

### 27.4.12 Starting a Scan Group by Using an A/D Timer Trigger

Scan group 3 or 4 can be started by using a trigger from A/D timer 3 or 4. To start scan group 3 or 4 by using a trigger from A/D timer 3 or 4, set TRGMD in ADCFnSGCR3 or ADCFnSGCR4 to $2_H$. Furthermore, set ADTST of scan group 3 or 4 to 1 to start A/D timer 3 or 4.

When a timer trigger is input in this state, SGACT is set to 1. The timing since SGACT is set to 1 until scan group 3 or 4 is started is the same as the timing when SGST is set to 1 by a software trigger.

### 27.4.13 Starting A/D Timer by Using a Hardware Trigger

A/D timer 3 or 4 can be started by using a hardware trigger SG3_TRG or SG4_TRG input. To start A/D timer 3 or 4 by using a hardware trigger SG3_TRG or SG4_TRG input, set TRGMD in ADCFnSGCR3 or ADCFnSGCR4 to $3_H$. When the selected external trigger is input in this state, A/D timer 3 or 4 starts. Furthermore, a trigger from A/D timer 3 or 4 starts scan group 3 or 4.

### 27.4.14 Monitoring Function by Using the A/D Conversion Monitor Pin

ADENDn can be used to monitor the processing timing of the virtual channel specified by ADCFnADENDP0.

**Figure 27.15** shows the A/D conversion monitor timing.

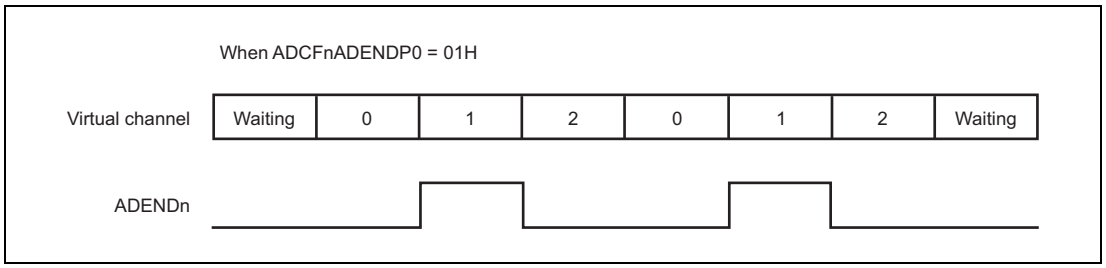



**Figure 27.15 A/D Conversion Monitor Timing**

**CAUTION**

**If the high-level voltage is output from ADENDn in a lower-priority scan group and a higher-priority scan group suspends (asynchronous suspend) the processing of the lower-priority scan group, the low-level voltage is output from ADENDn. Since the suspended virtual channel processing for the lower-priority scan group resumes after that, the high-level voltage is output again from ADENDn.**

### 27.4.15 Scan End Interrupt Request

Scan group x can issue a scan end interrupt request (ADInx) to the INTC. When ADIE in ADCFnSGCRx is set to 1, ADInx can be output after the SGx scan ends. When ADIE in ADCFnSGCRx is set to 0, the ADInx output at the end of the SGx scan can be disabled. When ADIE in ADCFnVCRj is set to 1, ADInx can be output when A/D conversion for virtual channel j in SGx ends. When ADIE in ADCFnVCRj is set to 0, the ADInx output at the end of the A/D conversion for virtual channel j in SGx can be disabled. The setting of ADIE in ADCFnSGCRx is independent of the setting of ADIE in ADCFnVCRj.

Example 1) A scan is executed for virtual channel 0 or 1 in SG0 when ADIE in ADCFnSGCR0 is 0, ADIE in ADCFnVCR0 is 1, and ADIE in ADCFnVCR1 is 0.
ADIn0 is output when A/D conversion ends for virtual channel 0.

Example 2) A scan is executed for virtual channel 0 or 1 in SG0 when ADIE in ADCFnSGCR0 is 0, ADIE in ADCFnVCR0 is 1, and ADIE in ADCFnVCR1 is 1.
ADIn0 is output when A/D conversion ends for virtual channel 0 and virtual channel 1.

Example 3) A scan is executed for virtual channel 0 or 1 in SG0 when ADIE in ADCFnSGCR0 is 1, ADIE in ADCFnVCR0 is 0, and ADIE in ADCFnVCR1 is 0.
ADIn0 is output when a scan ends (at the end of A/D conversion for virtual channel 1).
ADIn0 is output at each two cycles (or more) in multicycle scan mode, or each time a scan ends (at the end of A/D conversion for virtual channel 1) in continuous scan mode.

Furthermore, the DMAC can be activated when ADInx occurs.

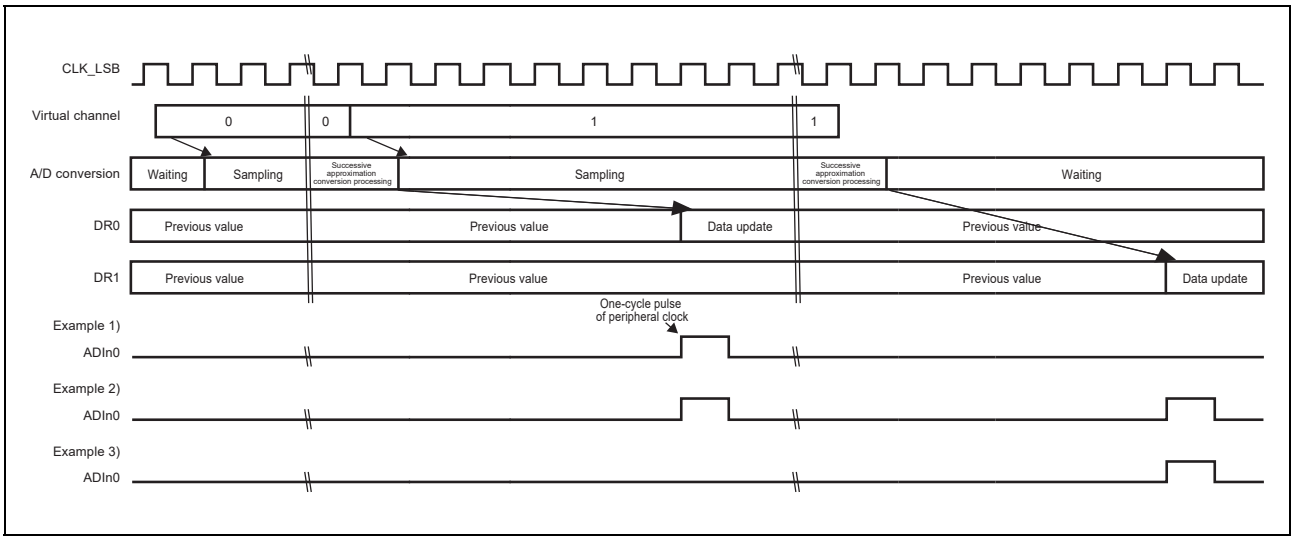



**Figure 27.16 Scan Conversion End Interrupt Occurrence Timing**

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

### 27.4.16 MPX Interrupt Request

The ADC can issue an MPX request (ADMPXIn) to the DMAC. ADMPXIn is generated when a virtual channel for which CNVCLS[2:0] in ADCFnVCRj is set to $5_H$ or $6_H$ is started.

The DMAC can be activated when ADMPXIn occurs.

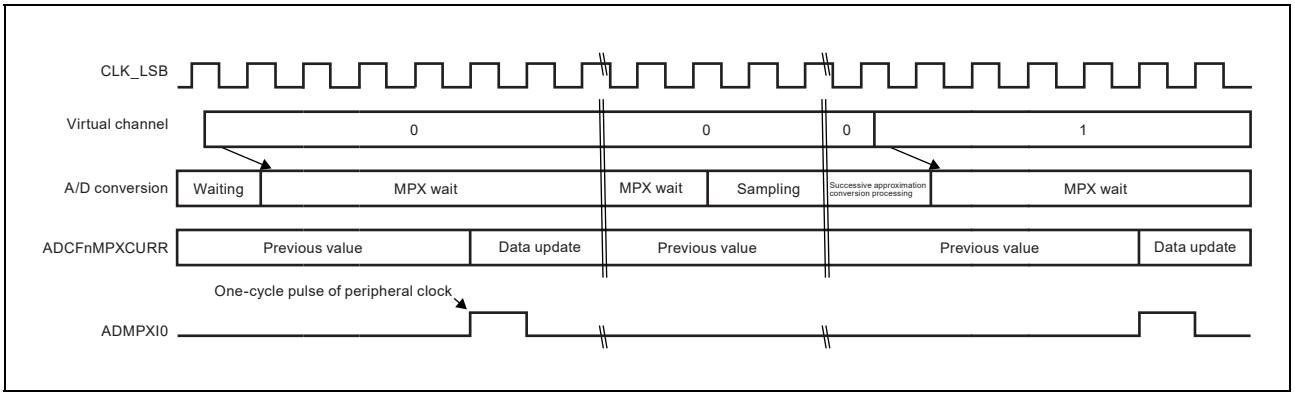



**Figure 27.17 Example of an MPX Interrupt Occurrence**

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

### 27.4.17 A/D Error Interrupt Request and A/D Parity Error Notification

The ADCF can issue an A/D error interrupt request (ADEm) to the INTC and an A/D parity error notification (ADPEm) to the error control module (ECM). For an error source for which ULEIE, OWEIE, and IDEIE in ADCFnSFTCR is set to 1, the OR condition of the error source is issued as ADEm. For an error source for which ULEIE, OWEIE, and IDEIE in ADCFnSFTCR is set to 0, ADEm can be disabled. ADPEm is enabled when PEIE in ADCFnSFTCR is set to 1. ADPEm is disabled when PEIE in ADCFnSFTCR is set to 0.

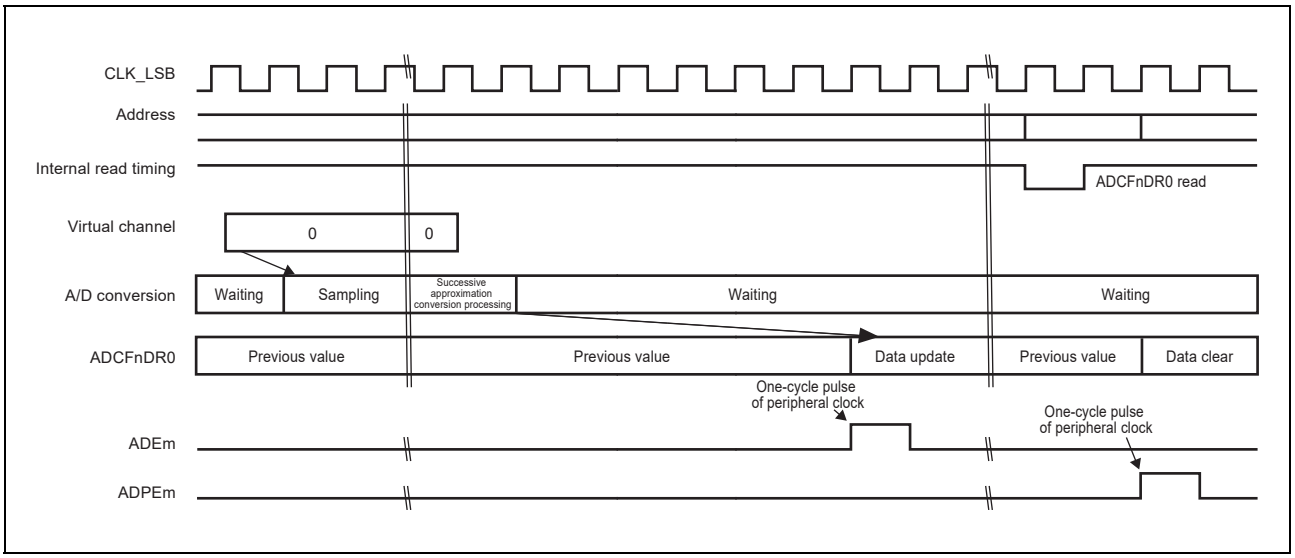



**Figure 27.18 Example of an Occurrence of A/D Error Interrupt and A/D Parity Error Notification**

**NOTE**

For the number of units and indices, see **Section 27.1.1, Number of Units**.

# 27.5 Operation

## 27.5.1 Initial Settings

AD conversion of the ADCF starts by setting the registers shown in **Figure 27.19**. For trigger input, see **Section 27.5.2, Trigger Input Flow**.

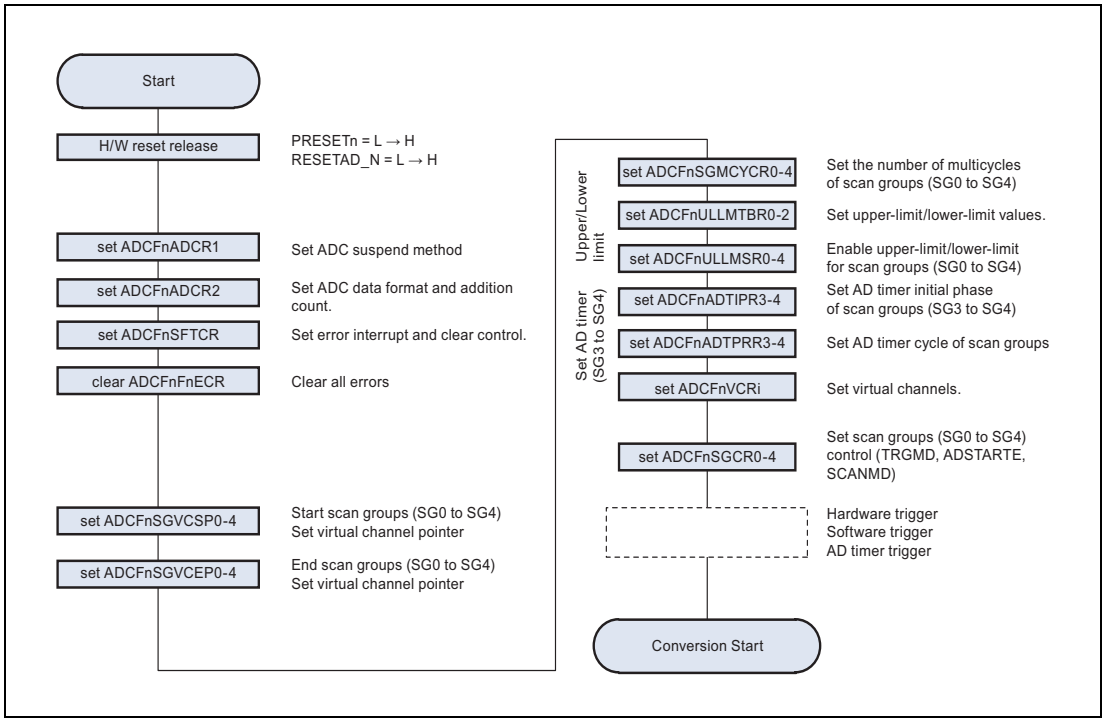



**Figure 27.19 Initial Settings**

### 27.5.2 Trigger Input Flow

A/D conversion start triggers of the ADCF include hardware triggers, software triggers, and A/D timer triggers. Scan group x (x = 0 to 4) supports the software trigger specified by ADSTART and SGST (SG0 to SG4), the hardware trigger of SGx_TRG (x = 0 to 4), and the A/D timer trigger (SG3 and SG4).

A/D conversion starts according to the trigger input flows shown in **Figure 27.20, Trigger Input Flow (SG0 to SG2)** and **Figure 27.21, Trigger Input Flow (SG3 to SG4)**, For the initial settings flow shown in these figures, see **Section 27.5.1, Initial Settings**.

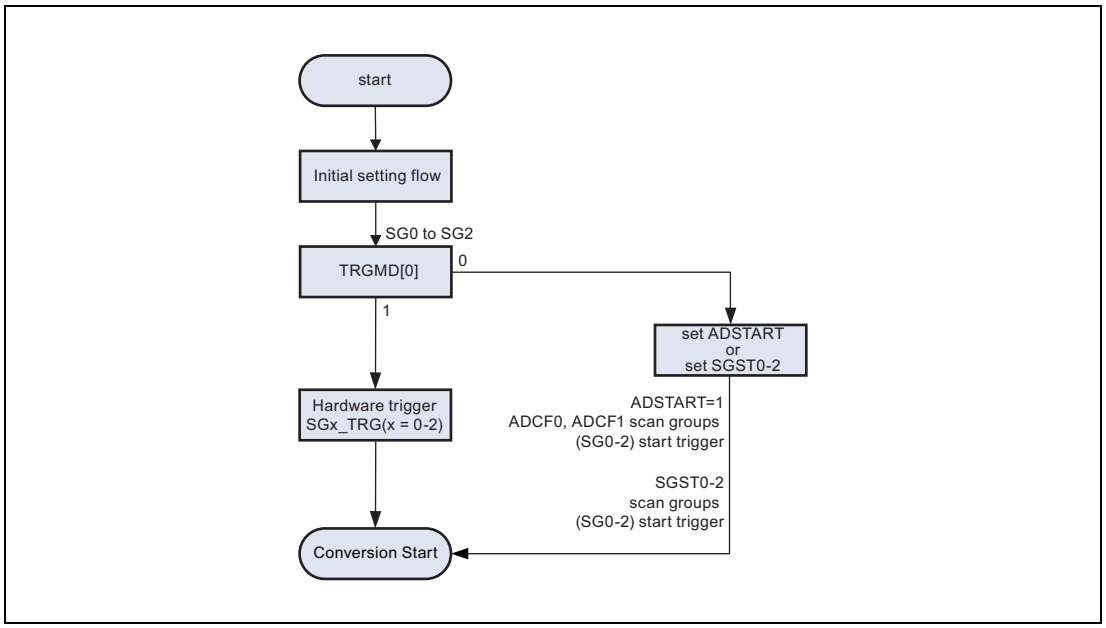



**Figure 27.20 Trigger Input Flow (SG0 to SG2)**

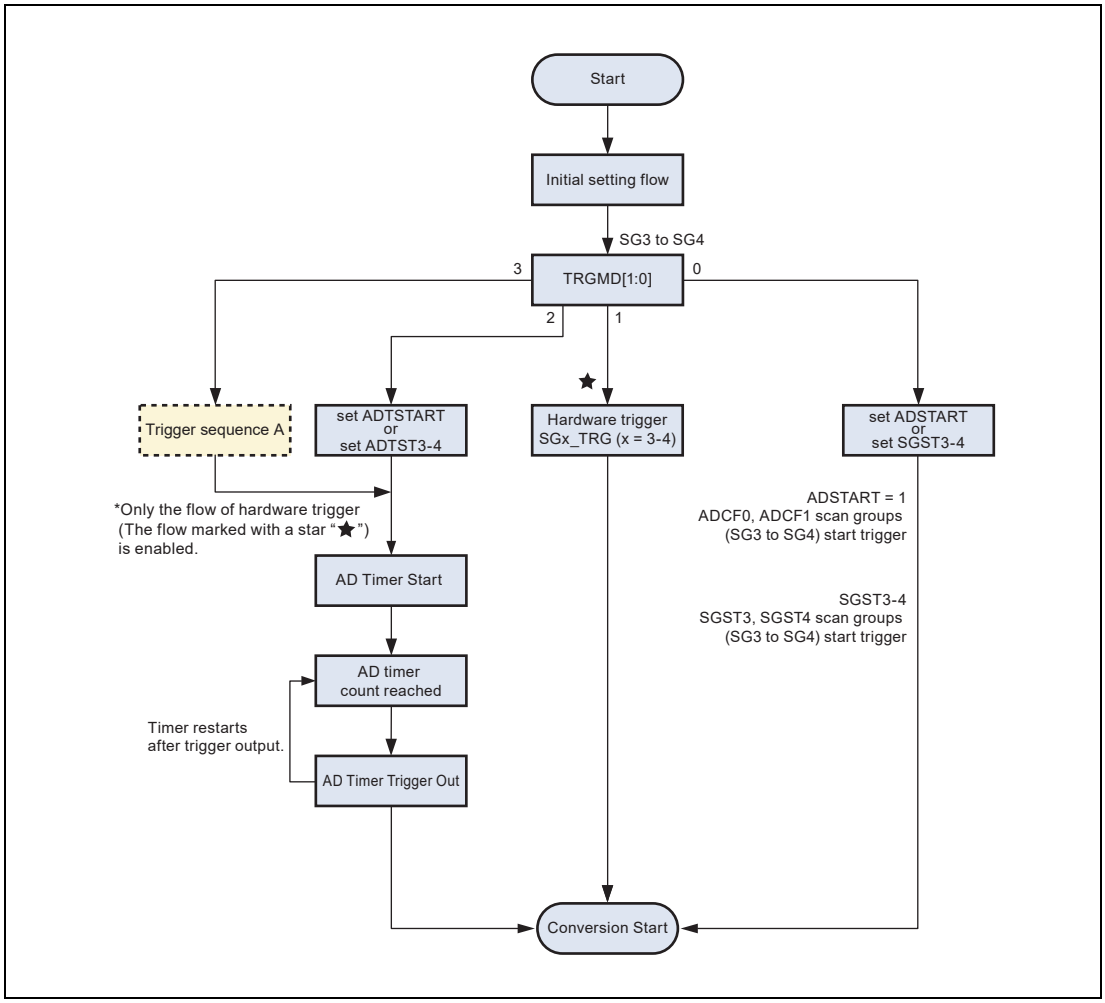



**Figure 27.21 Trigger Input Flow (SG3 to SG4)**

### 27.5.3 Terminating Procedure

The ADCF is forcibly terminated according to the processing flow shown in **Figure 27.22**.

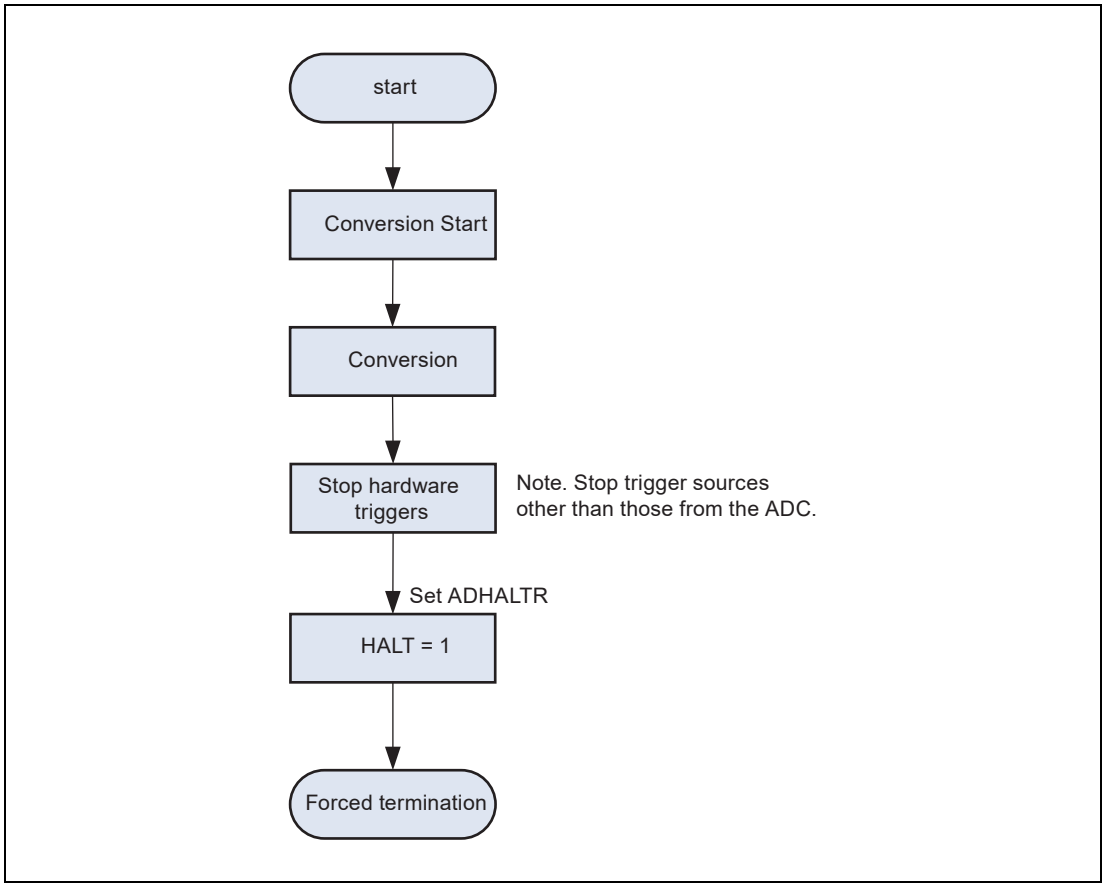



**Figure 27.22 Terminating Procedure**

### 27.5.4 Limited Reset and Module stand-by

The ADCF can be reset by limited reset from SYSCTRL. The Limited Reset is functionally equivalent to a hardware reset. Software must ensure that ADCF is halted. See **Section 27.5.3, Terminating Procedure**.

The ADCF clock can be disabled by the SYSCTRL module stand-by function. Software must ensure that ADCF is halted if module stand-by enable.

## 27.6 Definition of A/D Conversion Accuracy

A/D conversion accuracy is defined below.

- Resolution
  Number of digital output codes of the A/D converter
- Quantization error
  An error essentially contained in A/D converters, which is given as 1/2LSB (**Figure 27.23**).
- Offset error
  Deviation of the analog input voltage value from the ideal A/D conversion characteristics when the digital output changes from the minimum voltage value $000_H$ to $001_H$. However, the quantization error is not included (**Figure 27.23**).
- Full-scale error
  Deviation of the analog input voltage value from the ideal A/D conversion characteristics when the digital output changes from $FFE_H$ to $FFF_H$. However, the quantization error is not included (**Figure 27.23**)
- DNL (Differential nonlinear error)
  Deviation between the ideal digital output code width (Vq) and the actual digital output code width (Va), which is given as (Va – Vq)/Vq. However, the offset error, the full-scale error, and the quantization error are not included (**Figure 27.23**).
- INL (Integral nonlinear error)
  Deviation of the actual value from the ideal A/D conversion characteristics, from the zero voltage to the full-scale voltage, which is given as an integral of DNL from $000_H$ to a digital output code. However, the offset error, the full-scale error, and the quantization error are not included (**Figure 27.23**).
- Total overall error (TOE)
  Deviation between the digital value and the analog input value. The offset error, the full-scale error, the quantization error, DNL, and INL are included (**Figure 27.23**).

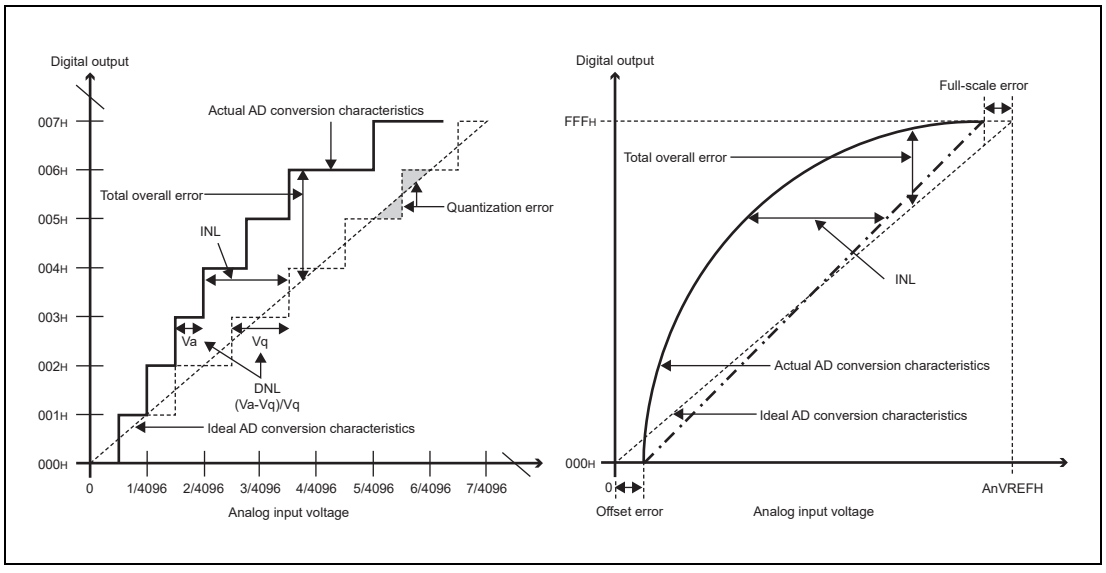



**Figure 27.23 Definition of A/D Conversion Accuracy**

## 27.7 Difference among P1M-C, P1H-C and P1H-CE

ADCFn module instance n = 0, 1 (refer to the table below for the different products)

- Physical channel index m
- Virtual channel index j

**Table 27.44 Different specifications of ADCF**

| Product | | P1M-C (QFP-144) | | P1M-C (BGA-292) | | P1M-C (BGA-156) | | P1H-C (4MB, BGA-156) | | P1H-C (4MB, BGA-292) | | P1H-C (8MB) and P1H-CE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ADCFn module** | | **ADCF0** | **ADCF1** | **ADCF0** | **ADCF1** | **ADCF0** | **ADCF1** | **ADCF0** | **ADCF1** | **ADCF0** | **ADCF1** | **ADCF0** | **ADCF1** |
| Physical channels | Total | 20 | | 24 | | 16 | | 16 | | 32 | | 40 | |
| | Channels/ ADCFm | 10 | 10 | 12 | 12 | 8 | 8 | 8 | 8 | 16 | 16 | 20 | 20 |
| | Index m | 00 to 09 | | 00 to 11 | | 00 to 07 | | 00 to 07 | | 00 to 15 | | 00 to 19 | |
| Virtual channels | Total | 44 | | 44 | | 44 | | 56 | | 56 | | 72 | |
| | Number/ ADCFm | 24 | 20 | 24 | 20 | 24 | 20 | 28 | 28 | 28 | 28 | 36 | 36 |
| | Index j | 00 to 23 | 00 to 19 | 00 to 23 | 00 to 19 | 00 to 23 | 00 to 19 | 00 to 27 | 00 to 27 | 00 to 27 | 00 to 27 | 00 to 35 | 00 to 35 |

# Section 28 Functional Safety

## 28.1 Overview

This chip is used for the automotive applications within a functional safety related context. Therefore the development according the relevant functional safety standard ISO26262 is considered. As this chip is intended for a broad range of chassis and non-chassis application with different environmental conditions and target functionality, the development is conducted based on a SEooC (Safety Element out of Context). Safety analysis based on ISO26262 such as FMEA, FTA and common cause failure analysis and safety implementations are conducted, and reports of each analysis and implemented measures will be provided to customer accordingly as ISO 26262 work products.

The 40-nm multi-core safety platform follows a structured approach as shown in basic safety architecture.

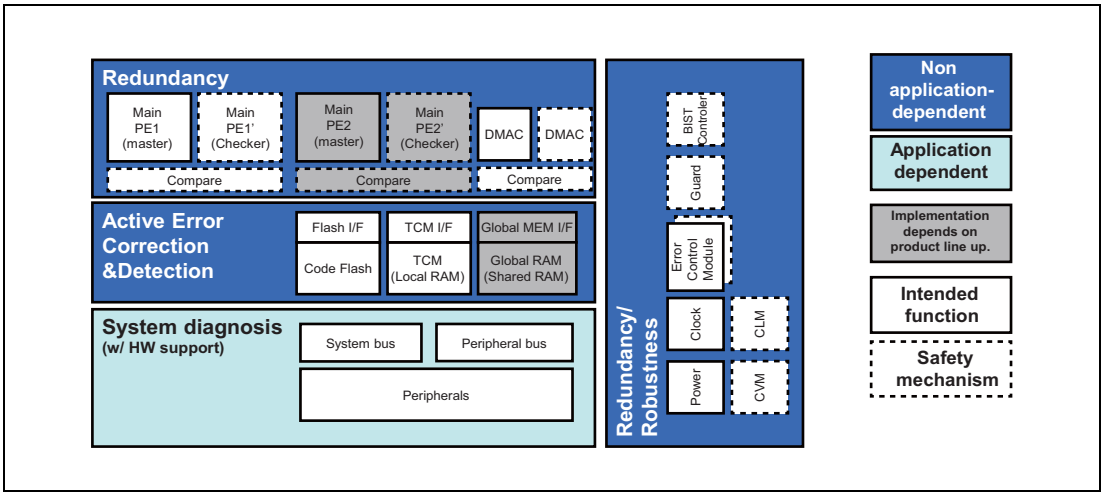



**Figure 28.1 Basic safety architecture.**

The platform can be categorized into application independent parts or dependent parts.

- Application independent parts includes the MCU parts
  - related to data processing i.e. CPU, DMA
  - related to data storage i.e. volatile and non-volatile memories
  - related to the MCU infrastructure common to all i.e. clock, reset and power supply
- Application dependent parts includes the MCU parts
  - related to interfaces to external. These are mainly the MCU peripherals, related busses and GPIO

P1M-C and P1H-C are developed according to safety plan which is compiled by product safety manager.

Based on MCU Technical Safety Concept and HW Safety Requirements point of view, Major safety mechanisms are listed as follows.

- Error Correction Code (ECC) and Error Detection Code (EDC)
- Lockstep Function of Redundant Data Processing Units (CPU, DMA) with compare units
- Memory Protection Unit (MPU) and Slave Guards for Processor Element (PEG), Internal Peripheral Modules (IPG), Global RAM (GRG) and Peripheral Bus (PBG)

- Field BIST at start-up test
- Error Control Module (ECM), for signaling error pin at failure detection by safety mechanisms
- Core Voltage Monitor (CVM)
- Clock Monitor (CLMA)
- Watch Dog Timer (WDTA)
- Data CRC (DCRB)
- Triple Modular Redundant (TMR) registers, for application independent parts to ensure robustness against transient faults caused by Single Event Effects (SEE)
- Safety Oriented Chip Layout

Safety work products can be referred to see how much these safety measures prevent risk from potential hardware faults. For details please contact your Safety Manager.

# 28.2 ECC and EDC

## 28.2.1 Overview

### 28.2.1.1 ECC

This product incorporates ECC for the following memories. The ECC enables detection and correction of errors of the data retained in the memories. The ECC also enables detection and correction of errors produced between the ECC encoder and memories; and memories and ECC decoder.

**Table 28.1 ECC Overview.**

| Applicable Memory | Applicable Data Width [bits] | Operation upon Error Detection | | | | |
|---|---|---|---|---|---|---|
| | | Detection/ Correction | Notice to ECM | Error Status | Address Capture | Failure Insertion |
| Code flash | 128 | SEC-DED | Possible | Possible | Possible | Possible |
| Data flash<br>Local RAM (CPU1, CPU2)<br>Global RAM | 32 | SEC-DED | Possible | Possible | Possible | Possible |
| Instruction cache (data) | 64 | SED-DED | Possible | Possible | Possible | Possible |
| Instruction cache (tag) | 32 | SED-DED | Possible | Possible | Possible | Possible |
| RAM for DTS | 32 | SEC-DED | Possible | Possible | Possible | Possible |
| Peripheral RAM (32 bits) | 32 | SEC-DED | Possible | Possible*1 | Possible*1 | Possible*1 |
| Data transfer path | 32 | SEC-DED | Possible | Possible | Possible | Possible |

Note 1. Except for Ethernet.

#### ECC code

ECC code with a Hamming distance of minimum 4 is used. Combination of data and ECC code space excludes all 0s or all 1s for RAM and code flash.

#### Detection/Correction

SEC-DED: 1-bit errors can be detected and corrected, and 2-bit errors can only be detected.

SED-DED: 1-bit errors and 2-bit errors can only be detected.

#### Notice to ECM

An error detected can be notified to the ECM (Error Control Module).

#### Error Status

The status of an error detected is retained.

#### Address Capture

The address of an error detected is retained.

#### Failure Insertion

An ECC error can be intentionally caused to enable self-diagnosis of the ECC decoder operation.

#### 28.2.1.2 Address Parity

This product incorporates address EDC (parity) for the following memories. The address EDC allows detection of errors during address decoding. The EDC also allows detection of errors produced at addresses between the parity encoder and memories.

**Table 28.2 Address Parity Overview.**

| Applicable Memory | Parity Bit | Notice to ECM | Error Status | Address Capture | Failure Insertion |
|---|---|---|---|---|---|
| Code flash | 1 bit | Possible | Possible | Possible | Possible |

## 28.2.2 Code Flash ECC and Address Parity

### 28.2.2.1 Overview

The code flash ECC for accesses from PE1, PE2 and over System Interconnect (VCI2CFB) is summarized in the table below.

**Table 28.3 Code Flash ECC**

| Item | Description |
|---|---|
| ECC error detection and correction | ECC error detection and correction can be either enabled or disabled.<br>When enabled, either of the following settings can be selected.<br>• ECC error detection and correction are carried out (2-bit error detection and 1-bit error detection and correction are carried out).<br>• ECC error detection is carried out (2-bit error detection and 1-bit error detection are carried out).<br>When disabled, neither error detection nor correction is carried out.<br>In the initial state, this function is enabled, and 1-bit error detection, correction and notification and 2-bit error detection and notification are carried out. |
| Address parity | Address parity check can be either enabled or disabled.<br>Address parity is checked during address decoding.<br>In the initial state, this function is enabled. |
| Error notification | Upon occurrence of an ECC error or parity error, it is notified to the Error Control Module.<br>ECC Error:<br>• Error notification can be either enabled or disabled upon detection of an ECC 2-bit error.<br>• Error notification can be either enabled or disabled upon detection of an ECC 1-bit error.<br>In the initial state, error notification is enabled upon detection of an ECC 2-bit error, and error notification is enabled upon detection of an ECC 1-bit error.<br>Parity Error:<br>• Error notification can be either enabled or disabled upon detection of an address parity error.<br>In the initial state, error notification is enabled upon detection of an address parity error.<br>Overflow Error:<br>• Error notification can be either enable or disable upon detection of an address buffer overflow error for ECC 1-bit error<br>In the initial state, error notification is enabled upon detection of an address buffer overflow error.<br>The error notification signal is issued to the ECM, where an ECC 2-bit error and an address parity error are handled as one source, and an ECC 1-bit error and overflow error are handled as one source, respectively. An ECC 1-bit error signal is always issued to the ECM, even when the ECC-1bit error address is already sotred in the error address buffer. |
| Error status | A status register is provided, which indicates the statuses of ECC 2-bit error detection, ECC 1-bit error detection, and address parity error detection. A four-stage buffer is provided for 1-bit errors. An one-stage buffer is provided for 2-bit error and the buffer is shared for parity error. The status register indicates the state of each stage.<br>The error status can be cleared using the clear register. |
| Address capture | The address is captured when an ECC 2-bit error, an ECC 1-bit error, or an address parity error is detected.<br>The error status serves as the enable bit of the capture address. |

### 28.2.2.2 List of Registers

Table 28.4 List of Registers

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 $2000_H$ | UCFERRINT | FLI (Code-Flash) Data and Address Error Information Control Register | R/W | 0000 $0047_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $2004_H$ | UCFSERSTCLR | FLI (Code-Flash) ECC SED Status Clear Register | W | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $2008_H$ | UCFDERSTCLR | FLI (Code-Flash) ECC DED/Address Parity Error Status Clear Register | W | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $200C_H$ | UCFOVFSTR | FLI (Code-Flash) Error Count Overflow Status Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $2020_H$ | UCFSERSTR | FLI(Code-Flash) ECC SED Status Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $2030_H$ | UCFDERSTR | FLI (Code-Flash) ECC DED/Address Parity Error Status Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $2040_H$ | UCF1SEDADR | FLI (Code-Flash) 1 ECC SED Address Register | R | 0000 $0000_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $2044_H$ | UCF2SEDADR | FLI (Code-Flash) 2 ECC SED Address Register | R | 0000 $0000_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $2048_H$ | UCF3SEDADR | FLI (Code-Flash) 3 ECC SED Address Register | R | 0000 $0000_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $204C_H$ | UCF4SEDADR | FLI (Code-Flash) 4 ECC SED Address Register | R | 0000 $0000_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $20C0_H$ | UCFDEDADR | FLI (Code-Flash) ECC DED/Address Parity Error Address Register | R | 0000 $0000_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $2100_H$ | CFAPCTL | FLI (Code-Flash) Address Parity Control Register | R/W | 0000 $0000_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $2200_H$ | CFECCCTL_VCI2CFBA | FLI (Code-Flash) ECC Control Register | R/W | 0000 $0000_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $22F0_H$ | CFSTSTCTL_VCI2CFBA | FLI(Code-Flash) Sub-Test Control Register | R/W | 0000 $0000_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $2400_H$ | CFECCCTL_PE1 | FLI(Code-Flash) ECC Control Register | R/W | 0000 $0000_H$ | 32/16 | APBGRD_ PFSS1.SP1 | — |
| FFC6 $24F0_H$ | CFSTSTCTL_PE1 | FLI(Code-Flash) Sub-Test Control Register | R/W | 0000 $0000_H$ | 32/16 | APBGRD_ PFSS1.SP1 | — |
| FFC6 $2600_H$ | CFECCCTL_PE2[*1] | FLI(Code-Flash) ECC Control Register | R/W | 0000 $0000_H$ | 32/16 | APBGRD_ PFSS1.SP2 | — |
| FFC6 $26F0_H$ | CFSTSTCTL_PE2[*1] | FLI(Code-Flash) Sub-Test Control Register | R/W | 0000 $0000_H$ | 32/16 | APBGRD_ PFSS1.SP2 | — |

Note 1. P1H-CE and P1H-C devices only. P1M-C device doesn't have these registers.

### 28.2.2.3 Details of Registers

#### (1) UCFERRINT — FLI (Code-Flash) Data and Address Error Information Control Register

UCFERRINT register controls whether error information is reported to ECM, when data ECC 2-bit error, data ECC 1-bit error, ECC 1-bit error overflow, and address parity error are detected by Code Flash access from PE1, PE2 or accesses over the System Interconnect to Code Flash.

**Access:** UCFERRINT can be read/written in 32/16/8-bit units.

**Address:** FFC6 $2000_H$

**Value after reset:** 0000 $0047_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | SEOVFIE | — | — | — | APEIE | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R/W | R/W | R/W |

**Table 28.5 UCFERRINT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 | SEOVFIE | ECC 1bit error overflow report enable bit<br>Overflow report control bit when 1bit error overflow flag (SERROVF) in UCFOVFSTR register is set.<br>0: ECC 1-bit error overflow report disabled<br>1: ECC 1-bit error overflow report enabled |
| 5 to 3 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 2 | APEIE | Address parity error report enable bit<br>Control error report of address parity error detection when address parity check is enabled.<br>0: Address parity error report disabled<br>1: Address parity error report enabled |
| 1 | DEDIE | ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when ECC error detection/correction is enabled<br>0: ECC 2-bit error report disabled<br>1: ECC 2-bit error report enabled |
| 0 | SEDIE | ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when ECC error detection/correction is enabled<br>0: ECC 1-bit error report disabled<br>1: ECC 1-bit error report enabled |

### (2) UCFSERSTCLR — FLI (Code-Flash) ECC SED Status Clear Register

UCFSERSTCLR register is used to clear SEDF[0:3] in UCFSERSTR, SERROVF in UCFOVFSTR, and error address in UCFnSEDADR (n = 1 to 4). This is write only register and read value is always “0”. UCFSERSTCLR register has a lower priority than a set factor. Priority is given to a set factor when UCFSERSTCLR and a set factor compete. A set factor means a trigger of setting SEDF[0:3] in UCFSERSTR, setting SERROVF in UCFOVFSTR, or capturing an error address in UCF[1:4]SEDADR.

**Access:** UCFSERSTCLR can be written only in 32/16/8-bit units.

**Address:** FFC6 $2004_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR3 | SSTCLR2 | SSTCLR1 | SSTCLR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | W | W | W | W |

**Table 28.6 UCFSERSTCLR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When writing, write the value after reset. |
| 3 to 0 | SSTCLR[3:0] | 1-bit error flag clear n th<br>1: All 1-bit error flag clear<br>UCFSERSTR.SEDFn,<br>UCFOVFSTR.SERROVF,<br>UCFnSEDADR.SEADR |

### (3) UCFDERSTCLR — FLI (Code-Flash) ECC DED/Address Parity Error Status Clear Register

UCFDERSTCLR register is used to clear DEDF and APEF in UCFDERSTR and error address in UCFDEDADR. This is write only register and read value is always “0”. UCFDERSTCLR register has a lower priority than a set factor. Priority is given to a set factor when UCFDERSTCLR and a set factor compete. A set factor means a trigger of setting APEF/DEDF in UCFDERSTR.

**Access:** UCFDERSTCLR can be written only in 32/16/8-bit units.

**Address:** FFC6 2008$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.7 UCFDERSTCLR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | DSTCLR | 1: All 2-bit error flag clear<br>UCFDERSTR.APEF,<br>UCFDERSTR.DEDF,<br>UCFDEDADR.DEADR |

### (4) UCFOVFSTR — FLI (Code-Flash) Error Count Overflow Status Register

UCFOVFSTR register monitor if error overflow occurs. Overflow occurs when a SED with the error address never captured is detected in the case that ECC 1-bit error status register is full. If the ECC 1-bit error status register is full and the error address is the same as the one of the error addresses already captured, this flag is not set. SERROVF flag is cleared by system reset, or cleared when at least one of SSTCLR[0:3] in UCFSERSTCLR register is asserted.

**Access:** UCFOVFSTR can be read in 32/16/8-bit units.

**Address:** FFC6 200C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SERR OVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.8 UCFOVFSTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SERROVF | 1-bit error overflow flag<br>This error flag is set if the followings occur.<br>• A SED occurrence when all bits (SEDF[0:3]) of ECC 1-bit error status register (UCFSERSTR) are set.<br>• If the ECC 1-bit error status register is full and the error address is the same as one of the error addresses already captured, this flag is not set. |

**Note:** As the overflow flag, only SED flags are prepared. Because DED is prepared for only one error address buffer, an overflow flag for DED is not necessary.

### (5) UCFSERSTR — FLI (Code-Flash) ECC SED Status Register

UCFSERSTR is the error monitor register for 1-bit ECC error. Each error flag is “0”and when a new error occurs, an error status flag is set. An error flag is set at the lowest number empty bit of UCFSERSTR (e.g. If SEDF[0][1][3] have been set to “1” and all other bits have been empty, a next flag is set to SEDF[2].). If multiple SED causes are detected simultaneously and there are empty bits sufficiently, detected SEDs are all set (e.g. if a SED and another SED which occurs at a different address are detected simultaneously, both SEDFs are set.). However, if SEDs occur at a same address simultaneously, only one of errors is set according to the fixed priority. The priority order is the PE1, PE2, and VCI2CFB, the higher priority it has. (e.g. if a SED which is input from the PE1 path and an SED which is input from the PE2 path are detected simultaneously and those occur at the same address, only SEDF from the PE1 path is set and SEDF from the PE2 path is not set.) And also, an error address which has already captured in UCFnSEDADR (n = 1 to 4) must not be captured again. UCFSERSTR register is cleared by system reset or SSTCLR[0:3] in UCFSERSTCLR register.

**Access:** UCFSERSTR can be read in 32/16/8-bit units.

**Address:** FFC6 $2020_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | SEDF3 | SEDF2 | SEDF1 | SEDF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.9 UCFSERSTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. |
| 3 to 0 | SEDF[3:0] | ECC 1-bit error monitor flag n th<br>Condition for “0”: system reset or write “1” to SSTCLRn in UCFSERSTCLR<br>Condition for “1”: ECC 1-bit error is detected. |

### (6) UCFDERSTR — FLI (Code-Flash) ECC DED/Address Parity Error Status Register

UCFDERSTR is the error monitor register for 2-bit ECC error and address parity error. All error flags are “0” and when a new error occurs, an error status flag is set. If multiple error causes which are input from each different slave port are detected simultaneously, only one of detected errors is set according to the fixed priority. The reason is that there is only one 2-bit error Address Register. The priority order is the PE1, PE2, and VCI2CFB. (e.g. if a DED which is input from the PE1 path and an APE which is input from the PE2 path are detected simultaneously, only DEDF from the PE1 path is set and APEF from the PE2 path is not set.) If multiple error causes occur from same slave port simultaneously, detected errors are all set. (e.g. if DED and APE are simultaneously input from the PE1 path, both DEDF and APEF are set.) UCFDERSTR register is cleared by system reset or DSTCLR in UCFDERSTCLR register.

**Access:** UCFDERSTR can be read in 32/16/8-bit units.

**Address:** FFC6 2030$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | APEF | — | DEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.10 UCFDERSTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is read. |
| 2 | APEF | Address parity error monitor flag<br>Condition to “0”: system reset or write “1” to DSTCLR in UCFDERSTCLR<br>Condition to “1”: APEF/DEDF is all “0” and address parity error is detected. |
| 1 | Reserved | When read, the value after reset is read. |
| 0 | DEDF | ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to DSTCLR in UCFDERCTCLR<br>Condition to “1”: DEDF is “0” and ECC 2-bit error is detected. |

### (7) UCFnSEDADR — FLI (Code-Flash) n ECC SED Address Register (n = 1 to 4)

UCFnSEDADR (n = 1 to 4) register is used to hold the address when an error is detected. Error address capture trigger is same as corresponding SEDF(n-1) set in UCFSERSTR. UCFnSEDADR (n = 1 to 4) is cleared by system reset or SSTCLR(n-1) in UCFSERSTCLR register.

**Access:** UCF*n*SEDADR can be read in 32-bit units.

**Address:** UCF1SEDADR: FFC6 2040$_H$
UCF2SEDADR: FFC6 2044$_H$
UCF3SEDADR: FFC6 2048$_H$
UCF4SEDADR: FFC6 204C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SEADR[24:16] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEADR[15:4] | | | | | | | | | | | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.11 UCFnSEDADR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 25 | Reserved | When read, the value after reset is read. |
| 24 to 4 | SEADR[24:4] | 1-bit error address |
| 3 to 0 | Reserved | When read, the value after reset is read. |

**NOTE**

P1M-C:

- FLI (Code-Flash)_Base_Address = 0x0000_0000
- Error address SED : ( FLI (Code-Flash)_Base_Address[31:25] , SEDARn[24:4] )

P1H-C (4MB), P1H-C (8MB):

- FLI (Code-Flash)_Base_Address = 0x0000_0000
- Error address SED : ( FLI (Code-Flash)_Base_Address[31:25] , SEDARn[24:4] )

P1H-CE (P1M-C mode), P1H-CE (P1H-C (4MB) mode), P1H-CE (P1H-C (8MB) mode):

- FLI (Code-Flash)_Base_Address = 0x0000_0000
- Error address SED : ( FLI (Code-Flash)_Base_Address[31:25] , SEDARn[24:4] )

### (8) UCFDEDADR — FLI (Code-Flash) ECC DED/Address Parity Error Address Register

UCFDEDADR register is used to hold the address when an error is detected. Error address capture trigger is same as corresponding APEF/DEDF set in UCFDERSTR. If this register has captured an error address, the register doesn't capture any more address before it is cleared. UCFDEDADR is cleared by system reset or DSTCLR in UCFDERSTCLR register.

**Access:** UCFDEDADR can be read in 32-bit units.

**Address:** FFC6 $20C0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DEADR[24:16] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEADR[15:4] | | | | | | | | | | | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.12 UCFDEDADR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 25 | Reserved | When read, the value after reset is read. |
| 24 to 4 | DEADR[24:4] | 2-bit error or address parity error address |
| 3 to 0 | Reserved | When read, the value after reset is read. |

**NOTE**

P1M-C:

- FLI (Code-Flash)_Base_Address = 0x0000_0000
- Error address DED : ( FLI (Code-Flash)_Base_Address[31:25] , DEADR[24:4] )

P1H-C (4MB), P1H-C (8MB):

- FLI (Code-Flash)_Base_Address = 0x0000_0000
- Error address DED : ( FLI (Code-Flash)_Base_Address[31:25] , DEADR[24:4] )

P1H-CE (P1M-C mode), P1H-CE (P1H-C (4MB) mode), P1H-CE (P1H-C (8MB) mode):

- FLI (Code-Flash)_Base_Address = 0x0000_0000
- Error address DED : ( FLI (Code-Flash)_Base_Address[31:25] , DEADR[24:4] )

### (9) CFAPCTL — FLI (Code-Flash) Address Parity Control Register

CFAPCTL register control the address parity check enable/disable. Writing address parity control registers must be executed with PROT[1:0] = $01_B$.

**Access:** CFAPCTL can be read/written in 32/16-bit units.

**Address:** FFC6 $2100_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT[1:0] | | — | — | — | — | — | — | — | — | — | — | — | APTESTB | APTESTA | APARIDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 28.13 CFAPCTL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 3 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 2 | APTESTB | Address Parity Check TEST Bit (Bank-B)<br>0: Address Parity Not Inverted (Normal Mode)<br>1: Address Parity Inverted (Test Mode) |
| 1 | APTESTA | Address Parity Check TEST Bit (Bank-A)<br>0: Address Parity Not Inverted (Normal Mode)<br>1: Address Parity Inverted (Test Mode) |
| 0 | APARIDIS | Address Parity disable bit<br>Setting Address Parity Check to enable/disable.<br>0: Address Parity Check is enable<br>1: Address Parity Check is disable |

### (10) CFECCCTL_VCI2CFBA/PE1/PE2 — FLI (Code-Flash) ECC Control Register

CFECCCTL_VCI2CFBA/PE1/PE2 register control the ECC error detection/correction and 1-bit error correction. Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

For the related access part, please consider this register:

CFECCCTL_VCI2CFBA: accesses over System Interconnect to Code Flash.

CFECCCTL_PE1 FLI: access from PE1

CFECCCTL_PE2 FLI: access from PE2.

**Access:** CFECCCTL_VCI2CFBA/PE1/PE2 can be read/written in 32/16-bit units.

**Address:**
CFECCCTL_VCI2CFBA: FFC6 $2200_H$
CFECCCTL_PE1: FFC6 $2400_H$
CFECCCTL_PE2: FFC6 $2600_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT[1:0] | | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.14 CFECCCTL_VCI2CFBA/PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SECDIS | ECC 1-bit error correction enable bit<br>When using ECC error detection/correction (ECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is done when 1-bit error is detected<br>1: No Correction is done when 1-bit error is detected |
| 0 | ECCDIS | ECC disable bit<br>Setting ECC error detection/correction to enable/disable.<br>0: ECC error detection/correction is enable<br>1: ECC error detection/correction is disable |

### (11) CFSTSTCTL_VCI2CFBA/PE1/PE2 — FLI (Code-Flash) Sub-Test Control Register

The CFSTSTCTL registers are used for the ECC test (self-diagnosis). These registers are dedicated for code flash. After ECC test mode is enabled by setting ECCTST = 1, the ECC bits and address parity bit can be read directly.

For the related access part, please consider this register:

CFSTSTCTL_VCI2CFBA: accesses over system interconnect to Code Flash.

CFSTSTCTL_PE1 FLI: access from PE1.

CFSTSTCTL_PE2 FLI: access from PE2.

**Access:** CFSTSTCTL_VCI2CFBA/PE1/PE2 can be read/written in 32/16-bit units.

**Address:** CFSTSTCTL_VCI2CFBA: FFC6 22F0$_H$
CFSTSTCTL_PE1: FFC6 24F0$_H$
CFSTSTCTL_PE2: FFC6 26F0$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT[1:0] | | — | — | — | — | — | — | — | — | — | — | — | — | — | ECCTST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 28.15 CFSTSTCTL_VCI2CFBA/PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | ECCTST | ECC test bit for Code Flash<br>By setting ECC test mode bit to “1” (ECCTST = 1), CPU can read ECC bit and address parity bit directly. Then read data have parity bit on bit 9 and ECC bits on bit 0 to 8. |

The CPU has a small data buffer. If an old value remains in this buffer, the correct value cannot be read even when the ECCTST bit is switched. When switching the ECCTST bit, be sure to clear the data buffer. For how to clear the data buffer, see **3.2.1.2 (7) Data Buffer Operation Function Registers**. This note is valid only for the CFSTSTCTL_PE1/PE2 register.

#### 28.2.2.4 Test Function

Through appropriate register setting, the code flash data, ECC bits, and address parity bit can be read out.

**(1) Reading code flash data**

(a) Set the ECCDIS bit in the code flash ECC control register to 1 to disable ECC error detection and correction.

(b) When ECCDIS = 1, neither error detection nor correction proceeds when the code flash is read; the data output from the code flash is read unchanged.

How to exit this test mode:

(a) Set the ECCDIS bit in the code flash ECC control register to 0 to enable ECC error detection and correction.

**(2) Reading the ECC and address parity bits**

(a) Set the ECCDIS bit in the code flash ECC control register to 1 to disable ECC error detection and correction.

(b) Set the ECCTST bit in the code flash sub-test control register to 1 to set test mode.

(c) When the code flash is read, the ECC and address parity bits are read instead of the code flash data.

How to exit this test mode:

(a) Set the ECCDIS bit in the code flash ECC control register to 0 to enable ECC error detection and correction.

(b) Set the ECCTST bit in the code flash sub-test control register to 0 to set normal mode.

**(3) Self-diagnosis of ECC check function**

Self-diagnosis of the ECC decoder for the access ports is possible by reading data from ECC test area. For the detail of ECC test area, refer **Section 32, Flash Memory**.

**(4) Self-diagnosis of address parity check function**

(a) Set the APTESTA (or APTESTB) bit in the code flash address parity control register to 1 to invert parity bit.

(b) When code flash data on any address in bank A (or bank B) is read, a fault can be injected to the address parity checker and self-diagnosis of the address parity check function is enabled

How to exit this test mode:

(a) Set the APTETA (or APTESTB) bit in the code flash address parity control register to 0 to set normal mode.

**NOTE**

P1H-C (4MB), P1H-C (8MB): The test code can be fetched from e.g. Local RAM.
P1M-C: The test code has to be fetched from e.g. Local RAM.
Otherwise, a pre-fetching during the test will cause the occurrence of SYSERR exception due to the inverted address parity bit.

## 28.2.3 Data Flash ECC

### 28.2.3.1 Overview

The data flash ECC is summarized in the table below.

**Table 28.16 Data Flash ECC**

| Item | Description |
|---|---|
| ECC error detection and correction | ECC error detection and correction can be either enabled or disabled.<br>When enabled, either of the following settings can be selected.<br>• ECC error detection and correction are carried out (2-bit error detection and 1-bit error detection and correction are carried out).<br>• ECC error detection is carried out (2-bit error detection and 1-bit error detection are carried out).<br>When disabled, neither error detection nor correction is carried out.<br>In the initial state, this function is enabled, and 1-bit error detection and correction and 2-bit error detection and notification are carried out. |
| Error notification | Upon occurrence of an ECC error, it is notified to the Error Control Module.<br>ECC Error:<br>• Error notification can be either enabled or disabled upon detection of an ECC 2-bit error.<br>• Error notification can be either enabled or disabled upon detection of an ECC 1-bit error.<br>In the initial state, error notification is enabled upon detection of an ECC 2-bit error, and error notification is disabled upon detection of an ECC 1-bit error.<br>Overflow Error:<br>• Error notification can be either enable or disable upon detection of an address buffer overflow error.<br>In the initial state, error notification is enabled upon detection of an address buffer overflow error.<br>The error notification signal is output, where an ECC 2-bit error, an ECC 1-bit error and an overflow error is handled as one source, respectively. |
| Error status | A status register is provided, which indicates the statuses of ECC 2-bit error detection and ECC 1-bit error detection. If an error occurs while no error status is set, the corresponding status is set. The error status can be cleared using the clear register. |
| Address capture | If an ECC error occurs while no error status is set, the address at which the associated error has occurred is captured.<br>The address is captured when an ECC 2-bit error or an ECC 1-bit error is detected. The error status serves as the enable bit of the capture address.<br>When error status is set and already latched ECC 1bit error or 2bit error occurs, error overflow is notified. |

### 28.2.3.2 List of Registers

Table 28.17 List of Registers

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC5 B000$_H$ | DFECCCTL0 | Data Flash ECC control register | R/W | 0000 0000$_H$ | 32/16 | — | — |
| FFC5 B004$_H$ | DFERSTR0 | Data Flash error status register | R | 0000 0000$_H$ | 32 | — | — |
| FFC5 B008$_H$ | DFERSTC0 | Data Flash error status clear register | W | 0000 0000$_H$ | 32/16/8 | — | — |
| FFC5 B00C$_H$ | DFOVFSTR0 | Data Flash error overflow status register | R | 0000 0000$_H$ | 32 | — | — |
| FFC5 B010$_H$ | DFOVFSTC0 | Data Flash error overflow status clear register | W | 0000 0000$_H$ | 32/16/8 | — | — |
| FFC5 B014$_H$ | DFERRINT0 | Data Flash error notification control register | R/W | 0000 0006$_H$ | 32/16/8 | — | — |
| FFC5 B018$_H$ | DFEADR0 | Data Flash 1st error address register | R | 0000 0000$_H$ | 32 | — | — |
| FFC5 B01C$_H$ | DFTSTCTL0 | Data flash test control register | R/W | 0000 0000$_H$ | 32/16 | — | — |
| FFC5 C000$_H$ | DFECCCTL1 | Data Flash ECC control register | R/W | 0000 0000$_H$ | 32/16 | — | — |
| FFC5 C004$_H$ | DFERSTR1 | Data Flash error status register | R | 0000 0000$_H$ | 32 | — | — |
| FFC5 C008$_H$ | DFERSTC1 | Data Flash error status clear register | W | 0000 0000$_H$ | 32/16/8 | — | — |
| FFC5 C00C$_H$ | DFOVFSTR1 | Data Flash error overflow status register | R | 0000 0000$_H$ | 32 | — | — |
| FFC5 C010$_H$ | DFOVFSTC1 | Data Flash error overflow status clear register | W | 0000 0000$_H$ | 32/16/8 | — | — |
| FFC5 C014$_H$ | DFERRINT1 | Data Flash error notification control register | R/W | 0000 0006$_H$ | 32/16/8 | — | — |
| FFC5 C018$_H$ | DFEADR1 | Data Flash 1st error address register | R | 0000 0000$_H$ | 32 | — | — |
| FFC5 C01C$_H$ | DFTSTCTL1 | Data flash test control register | R/W | 0000 0000$_H$ | 32/16 | — | — |

### 28.2.3.3 Details of Registers

#### (1) DFECCCTLn — Data flash ECC control register for n (bank number)

DFECCCTLn enables or disables ECC error detection and 1-bit error correction for read access to each data flash bank.

Set the PROT1 and PROT0 bits to $01_B$ when writing to DFECCCTLn.

**Access:** DFECCCTLn can be read/written in 32/16-bit units.

**Address:** DFECCCTL0: FFC5 $B000_H$
DFECCCTL1: FFC5 $C000_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.18 DFECCCTLn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 | PROT1 | Enables or disables modification of the ECCDIS and SECDIS bits. The value written is not retained. These bits are always read as 0. Set (PROT1, PROT0) = (0, 1) when writing to DFECCCTLn. |
| 14 | PROT0 | |
| 13 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SECDIS | 1-Bit Error Correction Disable<br>Enables or disables 1-bit error correction when ECC error detection and correction are enabled. Write a value to this bit simultaneously with setting (PROT1, PROT0) = (0, 1).<br>0: Enables correction of the 1-bit error detected.<br>1: Disables correction of the 1-bit error detected. |
| 0 | ECCDIS | ECC Disable<br>Enables or disables ECC error detection and correction. Write a value to this bit simultaneously with setting (PROT1, PROT0) = (0, 1).<br>In the initial state, ECC error detection and correction are enabled.<br>0: Enables ECC error detection and correction.<br>1: Disables ECC error detection and correction. |

### (2) DFERSTRn — Data flash error status register (n = 0, 1)

DFERSTRn monitors occurrence of errors.

The SEDF bit is set if an ECC 1-bit error is detected while ECC error detection and correction are enabled, and the DEDF bit is set if an ECC 2-bit error is detected.

**Access:** DFERSTRn can be read only in 32-bit units.

**Address:** DFERSTR0: FFC5 $B004_H$
DFERSTR1: FFC5 $C004_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDF | SEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.19 DFERSTRn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF | ECC 2-Bit Error Monitor Flag<br>0: An ECC 2-bit error is not generated.<br>1: An ECC 2-bit error is generated.<br><br>Clearing condition:<br>ERRCLR bit is set in Data Flash error status clear register.<br>Setting condition:<br>ECC 2-bit error is generated. |
| 0 | SEDF | ECC 1-bit error Monitor Flag<br>0: An ECC 1-bit error is not generated.<br>1: An ECC 1-bit error is generated.<br><br>Clearing condition:<br>ERRCLR bit is set in Data Flash error status clear register.<br>Setting condition:<br>ECC 1-bit error is generated with both SEDF and DEDF being 0. |

### (3) DFERSTCn — Data flash error status clear register (n = 0, 1)

DFERSTCn clears the error flags in the Data Flash error status register.

**Access:** DFERSTCn can be written only in 32/16/8-bit units.

**Address:** DFERSTC0: FFC5 $B008_H$
DFERSTC1: FFC5 $C008_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERRCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.20 DFERSTCn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | ERRCLR | SEDF/DEDF Flag Clear<br>0: No effect (Setting the ERRCLR bit to 0 does not affect the DEDF and SEDF flags in DFERSTRn.)<br>1: The SEDF/DEDF flag in DFERSTRn is cleared. |

### (4) DFOVFSTRn — Data flash error overflow status register (n = 0, 1)

DFOVFSTRn monitors occurrence of Data Flash error overflow.

**Access:** DFOVFSTRn can be read only in 32-bit units.

**Address:** DFOVFSTR0: FFC5 B00C$_H$
DFOVFSTR1: FFC5 C00C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERR OVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.21 DFOVFSTRn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ERROVF | Error Overflow Flag<br>ERROVF is set if the following occur<br>• An ECC 1-bit error is occurred when DFERSTRn .SEDF = 1 and access address is different from DFEADRn.<br>• An ECC 1-bit error is occurred when DFERSTRn .DEDF = 1<br>• An ECC 2-bit error is occurred when DFERSTRn .SEDF = 1<br>• An ECC 2-bit error is occurred when DFERSTRn .DEDF = 1 and access address is different from DFEADRn.<br><br>0: Not occurred<br>1: Occurred<br><br>Clearing condition: Set the ERROVFCLR bit in DFOVFSTC to 1. |

### (5) DFOVFSTCn — Data flash error overflow status clear register (n = 0, 1)

DFOVFSTCn clears the Data Flash error overflow flag.

**Access:** DFOVFSTCn can be written only in 32/16/8-bit units.

**Address:** DFOVFSTC0: FFC5B010$_H$
DFOVFSTC1: FFC5C010$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERR OVF CLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.22 DFOVFSTCn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | ERROVFCLR | Error Overflow Flag Clear<br>0: No effect (Setting the ERROVFCLR bit to 0 does not affect the flag in DFOVFSTR.)<br>1: The ERROVF flag in the DFOVFSTR register is cleared. |

### (6) DFERRINTn — Data flash error notification control register (n = 0, 1)

DFERRINTn enables or disables generation of the error notification signal upon detection of an address buffer overflow error, an ECC 2-bit error and an ECC 1-bit error.

**Access:** DFERRINTn can be read/written in 32/16/8-bit units.

**Address:** DFERRINT0: $FFC5B014_H$
DFERRINT1: $FFC5\ C014_H$

**Value after reset:** $0000\ 0006_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | EOVFIE | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

**Table 28.23 DFERRINTn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 2 | EOVFIE | ECC Error Address Overflow Notification Control<br>Enables or disables generation of the error notification signal upon detection of an address buffer overflow error.<br>0: Error address overflow report disabled<br>1: Error address overflow report enabled |
| 1 | DEDIE | ECC 2-bit error Notification Control<br>Enables or disables generation of the error notification signal upon detection of a 2-bit error when ECC error detection and correction are enabled.<br>0: Disables notification of the ECC 2-bit error.<br>1: Enables notification of the ECC 2-bit error. |
| 0 | SEDIE | ECC 1-bit error Notification Control<br>Enables or disables generation of the error notification signal upon detection of a 1-bit error when ECC error detection and correction are enabled.<br>0: Disables notification of the ECC 1-bit error.<br>1: Enables notification of the ECC 1-bit error. |

### (7) DFEADRn — Data flash 1st error address register (n = 0, 1)

DFEADRn holds the address if the following occur.

- An ECC 1bit error or 2 bit error occurred when DFERSTRn.SEDF = 0 and DFERSTRn.DEDF = 0
- An ECC 2bit error occurred when DFERSTRn.SEDF = 1 and DFERSTRn.DEDF = 0

**Access:** These registers can be read only in 32-bit units.

**Address:** DFEADR0: FFC5 $B018_H$
DFEADR1: FFC5 $C018_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | DFEAREA[1:0] | | DFEADR[18:16] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DFEADR[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.24 DFEADRn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. |
| 20, 19 | DFEAREA[1:0] | ECC Error Area<br>DFEAREA is a read-only field to monitor the area at which an ECC error has occurred.<br>This register holds an area information.<br>$0x_B$: User area. (base address BankA: FF20 $0000_H$, BankB: FF30 $0000_H$)<br>$10_B$: Configuration seting area. (Only Serial programming mode)<br>(base address BankA: FF28 $0000_H$, BankB: nothing)<br>$11_B$: OTP seting area. (Only Serial programming mode)<br>(base address BankA: FF2C $0000_H$, BankB: FF3C $0000_H$)<br>(OTP setting area is not required for Bank B in P1M-C)<br>Configuration and OTP setting areas are accessed under a specific condition relating to the setting of OTP or option byte. For the detail, see user's manual of flash library or flash memory hardware interface. |
| 18 to 2 | DFEADR[18:2] | ECC Error Address<br>DFEADR is a read-only field to monitor the address at which an ECC error has occurred.<br>This register holds an internal address. Convert it to the actual address by adding the data flash base address described in DFEAREA. |
| 1, 0 | Reserved | When read, the value after reset is read. |

### (8) DFTSTCTLn — Data flash test control register (n = 0, 1)

DFTSTCTLn is used for ECC testing.

The data of the ECC bit can be read after setting the ECC test mode (ECCTST = 1).

Set PROT[1:0] = 01 when writing to the DFTSTCTL register.

**Access:** DFTSTCTLn can be read/written in 32/16-bit units.

**Address:** DFTSTCTL0: FFC5 B01C$_H$
DFTSTCTL1: FFC5 C01C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | — | ECC TST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 28.25 DFTSTCTLn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 | PROT1 | Enables or disables modification of the ECCTST bit. The value written is not retained. These bits are always read as 0. Set (PROT1, PROT0) = (0, 1) when writing to DFECCCTL. |
| 14 | PROT0 | |
| 13 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | ECCTST | ECC Test<br>By setting ECC test mode bit to “1” (ECCTST = 1), CPU can read ECC bit.<br>Write a value to this bit simultaneously with setting (PROT1, PROT0) = (0, 1). |

#### 28.2.3.4 Test Function

Data in the ROM and the ECC bits can be read through the setting of the data flash test control register (DFTSTCTL).

**(1) Reading the ROM data**

(a) Set the ECCDIS bit in the data flash ECC control register to 1 to disable ECC error detection and correction.

(b) When ECCDIS = 1, neither error detection nor correction proceeds when the data flash is read; the data output from the data flash is read unchanged.

How to exit this test mode:

(a) Set the ECCDIS bit in the data flash ECC control register to 0 to enable ECC error detection and correction.

**(2) Reading the ECC bits**

(a) Set the ECCDIS bit in the data flash ECC control register to 1 to disable ECC error detection and correction.

(b) Set the ECCTST bit in the data flash control register to 1 to set test mode.

(c) When the data flash is read, the 7 lower-order bits of read data are read as ECC data.

How to exit this test mode:

(a) Set the ECCDIS bit in the data flash ECC control register to 0 to enable ECC error detection and correction.

(b) Set the ECCTST bit in the data flash test control register to 0 to set normal mode.

**(3) Self-diagnosis of ECC check function**

Self-diagnosis of the ECC decoder is possible by writing incorrect data to the data flash memory beforehand (fault injection) and then reading this data. A 1- or 2-bit ECC error fault can be injected by generating correct ECC bits once and inverting only the appropriate bits.

For details on programming of the data flash, see "*RH850/P1x-C Flash Memory User's Manual: Hardware Interface*".

## 28.2.4 Local RAM (CPU1/CPU2) ECC

### 28.2.4.1 Overview

Local RAM ECC of CPU1/CPU2 is summarized in the table below.

**Table 28.26 Local RAM ECC**

| Item | Description |
|---|---|
| ECC error detection and correction | ECC error detection and correction can be either enabled or disabled.<br>When enabled, either of the following settings can be selected.<br>• ECC error detection and correction are carried out (2-bit error detection and 1-bit error detection and correction are carried out).<br>• ECC error detection is carried out (2-bit error detection and 1-bit error detection are carried out).<br>When disabled, neither error detection nor correction is carried out.<br>In the initial state, the ECC function is enabled; 1-bit error detection, correction, and notification, and 2-bit error detection and notification are carried out. |
| Error notification | Upon occurrence of an ECC error, it is notified to the Error Control Module.<br><br>**ECC Error:**<br>• Error notification can be either enabled or disabled upon detection of an ECC 2-bit error.<br>• Error notification can be either enabled or disabled upon detection of an ECC 1-bit error.<br>In the initial state, notification of the 2-bit error is enabled and notification of the 1-bit error is enabled.<br><br>**Overflow Error:**<br>• Error notification can be either enable or disable upon detection of an address buffer overflow error for ECC 1-bit error<br>In the initial state, error notification is enabled upon detection of an address buffer overflow error.<br>The error notification signal is output, where an ECC 2-bit error, an ECC 1-bit error and an overflow error is handled as individual source. An ECC 1-bit error signal is only issued to the ECM, if the ECC 1-bit error address is not yet stored in the error address buffer. |
| Error status | A status register is provided, which indicates the statuses of ECC 2-bit error detection, ECC 1-bit error detection. If an error occurs while no error status is set, the corresponding status is set. The error status can be cleared using the clear register. |
| Address capture | Multi-stage address buffers are provided for an ECC 1bit error as similar to a code flash ECC.<br>1-bit error: Eight stages (in 32-bit units)<br>2-bit error: One stage (in 32-bit units)<br>The address is captured when an ECC 2-bit error or an ECC 1-bit error is detected. The error status serves as the enable bit of the capture address. |

### 28.2.4.2 List of Registers

Table 28.27 List of Registers (1/4)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 5400$_H$ | LRECCCTL_PE1 | Local-RAM ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5404$_H$ | LRERRINT_PE1 | Local RAM error information control register | R/W | 0000 0043$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5408$_H$ | LRSERSTCLR_PE1 | Local-RAM ECC SED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 540C$_H$ | LRDERSTCLR_PE1 | Local-RAM ECC DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5410$_H$ | LROVFSTR_PE1 | Local RAM error count overflow status register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5440$_H$ | LRSERSTR_PE1 | Local-RAM ECC SED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5450$_H$ | LRDERSTR_PE1 | Local-RAM ECC DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5460$_H$ | LR1SEDADR0_PE1 | Local RAM 1st error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5464$_H$ | LR1SEDADR1_PE1 | Local RAM 1st error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5468$_H$ | LR1SEDADR2_PE1 | Local RAM 1st error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 546C$_H$ | LR1SEDADR3_PE1 | Local RAM 1st error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5470$_H$ | LR2SEDADR0_PE1 | Local RAM 2nd error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5474$_H$ | LR2SEDADR1_PE1 | Local RAM 2nd error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5478$_H$ | LR2SEDADR2_PE1 | Local RAM 2nd error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 547C$_H$ | LR2SEDADR3_PE1 | Local RAM 2nd error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5480$_H$ | LR3SEDADR0_PE1 | Local RAM 3rd error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5484$_H$ | LR3SEDADR1_PE1 | Local RAM 3rd error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5488$_H$ | LR3SEDADR2_PE1 | Local RAM 3rd error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 548C$_H$ | LR3SEDADR3_PE1 | Local RAM 3rd error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5490$_H$ | LR4SEDADR0_PE1 | Local RAM 4th error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5494$_H$ | LR4SEDADR1_PE1 | Local RAM 4th error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5498$_H$ | LR4SEDADR2_PE1 | Local RAM 4th error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 549C$_H$ | LR4SEDADR3_PE1 | Local RAM 4th error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54A0$_H$ | LR5SEDADR0_PE1 | Local RAM 5th error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |

Table 28.27 List of Registers (2/4)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 54A4$_H$ | LR5SEDADR1_PE1 | Local RAM 5th error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54A8$_H$ | LR5SEDADR2_PE1 | Local RAM 5th error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54AC$_H$ | LR5SEDADR3_PE1 | Local RAM 5th error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54B0$_H$ | LR6SEDADR0_PE1 | Local RAM 6th error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54B4$_H$ | LR6SEDADR1_PE1 | Local RAM 6th error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54B8$_H$ | LR6SEDADR2_PE1 | Local RAM 6th error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54BC$_H$ | LR6SEDADR3_PE1 | Local RAM 6th error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54C0$_H$ | LR7SEDADR0_PE1 | Local RAM 7th error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54C4$_H$ | LR7SEDADR1_PE1 | Local RAM 7th error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54C8$_H$ | LR7SEDADR2_PE1 | Local RAM 7th error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54CC$_H$ | LR7SEDADR3_PE1 | Local RAM 7th error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54D0$_H$ | LR8SEDADR0_PE1 | Local RAM 8th error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54D4$_H$ | LR8SEDADR1_PE1 | Local RAM 8th error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54D8$_H$ | LR8SEDADR2_PE1 | Local RAM 8th error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54DC$_H$ | LR8SEDADR3_PE1 | Local RAM 8th error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54E0$_H$ | LRDEDADR0_PE1 | Local-RAM DED Address Register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54E4$_H$ | LRDEDADR1_PE1 | Local-RAM DED Address Register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54E8$_H$ | LRDEDADR2_PE1 | Local-RAM DED Address Register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 54EC$_H$ | LRDEDADR3_PE1 | Local-RAM DED Address Register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5604$_H$ | LRTSTCTL_PE1 | Local-RAM Test Control Register | R/W | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5608$_H$ | LRTDATBF0_PE1 | Local-RAM Test Data Read Buffer 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 560C$_H$ | LRTDATBF1_PE1 | Local-RAM Test Data Read Buffer 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 5800$_H$ | LRECCCTL_PE2[*1] | Local-RAM ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5804$_H$ | LRERRINT_PE2[*1] | Local RAM error information control register | R/W | 0000 0043$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5808$_H$ | LRSERSTCLR_PE2[*1] | Local-RAM ECC SED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |

Table 28.27 List of Registers (3/4)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 580C$_H$ | LRDERSTCLR_PE2[*1] | Local-RAM ECC DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5810$_H$ | LROVFSTR_PE2[*1] | Local RAM error count overflow status register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5840$_H$ | LRSERSTR_PE2[*1] | Local-RAM ECC SED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5850$_H$ | LRDERSTR_PE2[*1] | Local-RAM ECC DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5860$_H$ | LR1SEDADR0_PE2[*1] | Local RAM 1st error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5864$_H$ | LR1SEDADR1_PE2[*1] | Local RAM 1st error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5868$_H$ | LR1SEDADR2_PE2[*1] | Local RAM 1st error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 586C$_H$ | LR1SEDADR3_PE2[*1] | Local RAM 1st error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5870$_H$ | LR2SEDADR0_PE2[*1] | Local RAM 2nd error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5874$_H$ | LR2SEDADR1_PE2[*1] | Local RAM 2nd error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5878$_H$ | LR2SEDADR2_PE2[*1] | Local RAM 2nd error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 587C$_H$ | LR2SEDADR3_PE2[*1] | Local RAM 2nd error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5880$_H$ | LR3SEDADR0_PE2[*1] | Local RAM 3rd error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5884$_H$ | LR3SEDADR1_PE2[*1] | Local RAM 3rd error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5888$_H$ | LR3SEDADR2_PE2[*1] | Local RAM 3rd error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 588C$_H$ | LR3SEDADR3_PE2[*1] | Local RAM 3rd error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5890$_H$ | LR4SEDADR0_PE2[*1] | Local RAM 4th error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5894$_H$ | LR4SEDADR1_PE2[*1] | Local RAM 4th error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5898$_H$ | LR4SEDADR2_PE2[*1] | Local RAM 4th error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 589C$_H$ | LR4SEDADR3_PE2[*1] | Local RAM 4th error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58A0$_H$ | LR5SEDADR0_PE2[*1] | Local RAM 5th error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58A4$_H$ | LR5SEDADR1_PE2[*1] | Local RAM 5th error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58A8$_H$ | LR5SEDADR2_PE2[*1] | Local RAM 5th error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58AC$_H$ | LR5SEDADR3_PE2[*1] | Local RAM 5th error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58B0$_H$ | LR6SEDADR0_PE2[*1] | Local RAM 6th error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58B4$_H$ | LR6SEDADR1_PE2[*1] | Local RAM 6th error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |

Table 28.27 List of Registers (4/4)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 58B8$_H$ | LR6SEDADR2_PE2[*1] | Local RAM 6th error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58BC$_H$ | LR6SEDADR3_PE2[*1] | Local RAM 6th error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58C0$_H$ | LR7SEDADR0_PE2[*1] | Local RAM 7th error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58C4$_H$ | LR7SEDADR1_PE2[*1*1] | Local RAM 7th error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58C8$_H$ | LR7SEDADR2_PE2[*1] | Local RAM 7th error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58CC$_H$ | LR7SEDADR3_PE2[*1] | Local RAM 7th error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58D0$_H$ | LR8SEDADR0_PE2[*1] | Local RAM 8th error address register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58D4$_H$ | LR8SEDADR1_PE2[*1] | Local RAM 8th error address register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58D8$_H$ | LR8SEDADR2_PE2[*1] | Local RAM 8th error address register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58DC$_H$ | LR8SEDADR3_PE2[*1] | Local RAM 8th error address register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58E0$_H$ | LRDEDADR0_PE2[*1] | Local-RAM DED Address Register 0 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58E4$_H$ | LRDEDADR1_PE2[*1] | Local-RAM DED Address Register 1 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58E8$_H$ | LRDEDADR2_PE2[*1] | Local-RAM DED Address Register 2 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 58EC$_H$ | LRDEDADR3_PE2[*1] | Local-RAM DED Address Register 3 | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5A04$_H$ | LRTSTCTL_PE2[*1] | Local-RAM Test Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5A08$_H$ | LRTDATBF0_PE2[*1] | Local-RAM Test Data Read Buffer 0 | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 5A0C$_H$ | LRTDATBF1_PE2[*1] | Local-RAM Test Data Read Buffer 1 | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |

Note 1. P1H-CE and P1H-C devices only. P1M-C device doesn't have these registers.

### 28.2.4.3 Details of Registers

#### (1) LRECCCTL_PE1/PE2 — Local-RAM ECC Control Register

LRECCCTL_PE1/PE2 register control the ECC error detection/correction and 1-bit error correction. Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

**Access:** LRECCCTL_PE1/PE2 can be read/written in 32/16-bit units.

**Address:** LRECCCTL_PE1: FFC6 $5400_H$
LRECCCTL_PE2: FFC6 $5800_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT[1:0] | | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.28 LRECCCTL_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SECDIS | ECC 1-bit error correction enable bit<br>When using ECC error detection/correction (ECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is carried out when 1-bit is detected<br>1: Correction is not carried out when 1-bit is detected |
| 0 | ECCDIS | ECC disable bit<br>Setting ECC error detection/correction to enable/disable.<br>0: ECC error detection/correction is enable<br>1: ECC error detection/correction is disable |

### (2) LRERRINT_PE1/PE2 — Local-RAM Error Information Control Register

LRERRINT_PE1/PE2 register controls whether error information is reported to ECM, when data ECC 2-bit error, ECC 1 bit error overflow and data ECC 1-bit error are detected.

**Access:** LRERRINT_PE1/PE2 can be read/written in 32/16/8-bit units.

**Address:** LRERRINT_PE1: FFC6 5404$_H$
LRERRINT_PE2: FFC6 5804$_H$

**Value after reset:** 0000 0043$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | EOVFIE | — | — | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R | R/W | R/W |

**Table 28.29 LRERRINT_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 | EOVFIE | ECC 1bit error overflow report enable bit<br>Overflow report control bit when 1bit error overflow flag (SERROVFn) in LROVFSTR register is set.<br>0: ECC 1bit error overflow report disabled<br>1: ECC 1bit error overflow report enabled |
| 5 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when ECC error detection/correction is enabled<br>0: ECC 2-bit error report disabled<br>1: ECC 2-bit error report enabled |
| 0 | SEDIE | ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when ECC error detection/correction is enabled<br>0: ECC 1-bit error report disabled<br>1: ECC 1-bit error report enabled |

### (3) LRSERSTCLR_PE1/PE2 — Local-RAM ECC SED Status Clear Register

LRSERSTCLR_PE1/PE2 register is used to clear SEDFmn (m = 1 to 8, n = 0 to 3) in LRSERSTR_PE1/PE2 and SERROVFn (n = 0 to 3) in LROVFSTR_PE1/PE2, and error address in LRmSEDADRn_PE1/PE2. This is write only register and read value is always “0”.

**Access:** LRSERSTCLR_PE1/PE2 can be written in 32/16/8-bit units.

**Address:** LRSERSTCLR_PE1: FFC6 5408$_H$
LRSERSTCLR_PE2: FFC6 5808$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSTCLR83 | SSTCLR82 | SSTCLR81 | SSTCLR80 | SSTCLR73 | SSTCLR72 | SSTCLR71 | SSTCLR70 | SSTCLR63 | SSTCLR62 | SSTCLR61 | SSTCLR60 | SSTCLR53 | SSTCLR52 | SSTCLR51 | SSTCLR50 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSTCLR43 | SSTCLR42 | SSTCLR41 | SSTCLR40 | SSTCLR33 | SSTCLR32 | SSTCLR31 | SSTCLR30 | SSTCLR23 | SSTCLR22 | SSTCLR21 | SSTCLR20 | SSTCLR13 | SSTCLR12 | SSTCLR11 | SSTCLR10 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 28.30 LRSERSTCLR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SSTCLRmn (m = 1 to 8) (n = 0 to 3) | 1-bit error flag clear (bank n)<br>1: All 1-bit error flag clear<br>LRSERSTR_PE1/PE2.SEDFmn, LROVFSTR_PE1/PE2.SERROVFn, LRmSEDADRn_PE1/PE2.SEADR |

### (4) LRDERSTCLR_PE1/PE2 — Local-RAM ECC DED Status Clear Register

LRDERSTCLR_PE1/PE2 register is used to clear DEDFn (n = 0 to 3) in LRDERSTR_PE1/PE2 and DERROVFn (n=0 to 3) in LROVFSTR_PE1/PE2 and error address in LRDEDADRn_PE1/PE2 (n = 0 to 3). This is write only register and read value is always “0”.

**Access:** LRDERSTCLR_PE1/PE2 can be written in 32/16/8-bit units.

**Address:** LRDERSTCLR_PE1: FFC6 540C$_H$
LRDERSTCLR_PE2: FFC6 580C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | DSTCLR3 | DSTCLR2 | DSTCLR1 | DSTCLR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | W | W | W | W |

**Table 28.31 LRDERSTCLR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When writing, write the value after reset. |
| 3 to 0 | DSTCLR[3:0] | 2-bit error flag clear (bank n)<br>1: All 2-bit error flag clear<br>LRDERSTR_PE1/PE2.DEDFn,<br>LROVFSTR_PE1/PE2.DERROVFn<br>LRDEDADRn_PE1/PE2.DEADR |

### (5) LROVFSTR_PE1/PE2 — Local-RAM Error Count Overflow Status Register

LROVFSTR_PE1/PE2 register monitor if error overflow occurs. Overflow occurs when different overflow*1 is detected in the case that error status is full. If the error status is full and the same error (same error cause and same error address) has occurred, then this flag is not set. SERROVF*n* (*n* = 0 to 3) , DERROVFn (n=0 to 3) flag is cleared by system reset or SSTCLR*mn* (*m* = 1 to 8, *n* = 0 to 3) ,DSTCLRn (n=0-3) in LRSERSTCLR_PE1/PE2 , LRDERSTCLR_PE1/PE2 register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** LROVFSTR_PE1/PE2 can be read in 32/16/8-bit units.

**Address:** LROVFSTR_PE1: FFC6 5410$_H$
LROVFSTR_PE2: FFC6 5810$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DERROVF3 | DERROVF2 | DERROVF1 | DERROVF0 | SERROVF3 | SERROVF2 | SERROVF1 | SERROVF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.32 LROVFSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is read. |
| 7 to 4 | DERROVF[3:0] | 2-bit error overflow flag (bankn)<br>This error flag is set if the followings occur.<br>• DEDFn in LRDERSTR_PE1/PE2 is set.<br>• Error with neither the same error cause nor the same address occur. |
| 3 to 0 | SERROVF[3:0] | 1-bit error overflow flag (bankn)<br>This error flag is set if the followings occur.<br>• SEDFn in LRSERSTR_PE1/PE2 is set.<br>• Error with neither the same error cause nor the same address occur. |

### (6) LRSERSTR_PE1/PE2 — Local-RAM ECC SED Status Register

LRSERSTR_PE1/PE2 is the error monitor register. If the error flag is “0” for each bank and a new error occurs, the error status flag is set. LRSERSTR_PE1/PE2 register is cleared by system reset or SSTCLR*mn* (*m* = 1 to 8, *n* = 0 to 3) in LRSERSTCLR_PE1/PE2 register.

**Access:** LRSERSTR_PE1/PE2 can be read in 32/16/8-bit units.

**Address:** LRSERSTR_PE1: FFC6 5440$_H$
LRSERSTR_PE2: FFC6 5840$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEDF83 | SEDF82 | SEDF81 | SEDF80 | SEDF73 | SEDF72 | SEDF71 | SEDF70 | SEDF63 | SEDF62 | SEDF61 | SEDF60 | SEDF53 | SEDF52 | SEDF51 | SEDF50 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEDF43 | SEDF42 | SEDF41 | SEDF40 | SEDF33 | SEDF32 | SEDF31 | SEDF30 | SEDF23 | SEDF22 | SEDF21 | SEDF20 | SEDF13 | SEDF12 | SEDF11 | SEDF10 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.33 LRSERSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SEDF*mn*<br>(*m* = 1 to 8)<br>(*n* = 0 to 3) | ECC 1-bit error monitor flag (bank n)<br>Condition to “0”: system reset or write “1” to SSTCLR*mn* in LRSERSTCLR_PE1/PE2<br>Condition to “1”: SEDF*mn* is “0” and ECC 1-bit error is detected. |

### (7) LRDERSTR_PE1/PE2 — Local-RAM ECC DED Status Register

LRDERSTR_PE1/PE2 is the error monitor register. If an error flag is “0” for a specific bank and a new ECC 2-bit error occurs by reading from this bank, then the error status flag is set. LRDERSTR_PE1/PE2 register is cleared by system reset or DSTCLRn (n = 0 to 3) in LRDERSTCLR_PE1/PE2 register.

**Access:** LRDERSTR_PE1/PE2 can be read in 32/16/8-bit units.

**Address:** LRDERSTR_PE1: FFC6 5450$_H$
LRDERSTR_PE2: FFC6 5850$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | DEDF3 | — | — | — | — | — | — | — | DEDF2 | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | DEDF1 | — | — | — | — | — | — | — | DEDF0 | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.34 LRDERSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | When read, the value after reset is read. |
| 25 | DEDF3 | ECC 2-bit error monitor flag (bank 3)<br>Condition to “0”: system reset or write “1” to DSTCLR3 in LRDERSTCLR_PE1/PE2<br>Condition to “1”: DEDF3 is “0” and ECC 2-bit error is detected. |
| 24 to 18 | Reserved | When read, the value after reset is read. |
| 17 | DEDF2 | ECC 2-bit error monitor flag (bank 2)<br>Condition to “0”: system reset or write “1” to DSTCLR2 in LRDERSTCLR_PE1/PE2<br>Condition to “1”: DEDF2 is “0” and ECC 2-bit error is detected. |
| 16 to 10 | Reserved | When read, the value after reset is read. |
| 9 | DEDF1 | ECC 2-bit error monitor flag (bank 1)<br>Condition to “0”: system reset or write “1” to DSTCLR1 in LRDERSTCLR_PE1/PE2<br>Condition to “1”: DEDF1 is “0” and ECC 2-bit error is detected. |
| 8 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF0 | ECC 2-bit error monitor flag (bank 0)<br>Condition to “0”: system reset or write “1” to DSTCLR0 in LRDERSTCLR_PE1/PE2<br>Condition to “1”: DEDF0 is “0” and ECC 2-bit error is detected. |
| 0 | — | Reserved. These bits are always read as 0. The write value should also be 0. |

### (8) LR*m*SEDADR*n*_PE1/PE2 — Local-RAM 1st to 8th SED Address Register *n* (*m* = 1 to 8, *n* = 0 to 3)

LR*m*SEDADR*n*_PE1/PE2 (*m* = 1 to 8, *n* = 0 to 3) register is used to hold the address when an error is detected. Error address capture trigger is same as corresponding SEDF*mn* (*m* = 1 to 8, *n* = 0 to 3) set in LRSERSTR_PE1/PE2. LR*m*SEDADR*n*_PE1/PE2 (*m* = 1 to 8, *n* = 0 to 3) is cleared by system reset or SSTCLR*mn* (*m* = 1 to 8, *n* = 0 to 3) in LRSERSTCLR_PE1/PE2 register.

**Access:** LR*m*SEDADR*n*_PE1/PE2 can be read in 32-bit units.

**Address:**

| | |
|---|---|
| LR1SEDADR0_PE1: FFC6 5460$_H$ | LR1SEDADR1_PE1: FFC6 5464$_H$ |
| LR1SEDADR2_PE1: FFC6 5468$_H$ | LR1SEDADR3_PE1: FFC6 546C$_H$ |
| LR2SEDADR0_PE1: FFC6 5470$_H$ | LR2SEDADR1_PE1: FFC6 5474$_H$ |
| LR2SEDADR2_PE1: FFC6 5478$_H$ | LR2SEDADR3_PE1: FFC6 547C$_H$ |
| LR3SEDADR0_PE1: FFC6 5480$_H$ | LR3SEDADR1_PE1: FFC6 5484$_H$ |
| LR3SEDADR2_PE1: FFC6 5488$_H$ | LR3SEDADR3_PE1: FFC6 548C$_H$ |
| LR4SEDADR0_PE1: FFC6 5490$_H$ | LR4SEDADR1_PE1: FFC6 5494$_H$ |
| LR4SEDADR2_PE1: FFC6 5498$_H$ | LR4SEDADR3_PE1: FFC6 549C$_H$ |
| LR5SEDADR0_PE1: FFC6 54A0$_H$ | LR5SEDADR1_PE1: FFC6 54A4$_H$ |
| LR5SEDADR2_PE1: FFC6 54A8$_H$ | LR5SEDADR3_PE1: FFC6 54AC$_H$ |
| LR6SEDADR0_PE1: FFC6 54B0$_H$ | LR6SEDADR1_PE1: FFC6 54B4$_H$ |
| LR6SEDADR2_PE1: FFC6 54B8$_H$ | LR6SEDADR3_PE1: FFC6 54BC$_H$ |
| LR7SEDADR0_PE1: FFC6 54C0$_H$ | LR7SEDADR1_PE1: FFC6 54C4$_H$ |
| LR7SEDADR2_PE1: FFC6 54C8$_H$ | LR7SEDADR3_PE1: FFC6 54CC$_H$ |
| LR8SEDADR0_PE1: FFC6 54D0$_H$ | LR8SEDADR1_PE1: FFC6 54D4$_H$ |
| LR8SEDADR2_PE1: FFC6 54D8$_H$ | LR8SEDADR3_PE1: FFC6 54DC$_H$ |
| LR1SEDADR0_PE2: FFC6 5860$_H$ | LR1SEDADR1_PE2: FFC6 5864$_H$ |
| LR1SEDADR2_PE2: FFC6 5868$_H$ | LR1SEDADR3_PE2: FFC6 586C$_H$ |
| LR2SEDADR0_PE2: FFC6 5870$_H$ | LR2SEDADR1_PE2: FFC6 5874$_H$ |
| LR2SEDADR2_PE2: FFC6 5878$_H$ | LR2SEDADR3_PE2: FFC6 587C$_H$ |
| LR3SEDADR0_PE2: FFC6 5880$_H$ | LR3SEDADR1_PE2: FFC6 5884$_H$ |
| LR3SEDADR2_PE2: FFC6 5888$_H$ | LR3SEDADR3_PE2: FFC6 588C$_H$ |
| LR4SEDADR0_PE2: FFC6 5890$_H$ | LR4SEDADR1_PE2: FFC6 5894$_H$ |
| LR4SEDADR2_PE2: FFC6 5898$_H$ | LR4SEDADR3_PE2: FFC6 589C$_H$ |
| LR5SEDADR0_PE2: FFC6 58A0$_H$ | LR5SEDADR1_PE2: FFC6 58A4$_H$ |
| LR5SEDADR2_PE2: FFC6 58A8$_H$ | LR5SEDADR3_PE2: FFC6 58AC$_H$ |
| LR6SEDADR0_PE2: FFC6 58B0$_H$ | LR6SEDADR1_PE2: FFC6 58B4$_H$ |
| LR6SEDADR2_PE2: FFC6 58B8$_H$ | LR6SEDADR3_PE2: FFC6 58BC$_H$ |
| LR7SEDADR0_PE2: FFC6 58C0$_H$ | LR7SEDADR1_PE2: FFC6 58C4$_H$ |
| LR7SEDADR2_PE2: FFC6 58C8$_H$ | LR7SEDADR3_PE2: FFC6 58CC$_H$ |
| LR8SEDADR0_PE2: FFC6 58D0$_H$ | LR8SEDADR1_PE2: FFC6 58D4$_H$ |
| LR8SEDADR2_PE2: FFC6 58D8$_H$ | LR8SEDADR3_PE2: FFC6 58DC$_H$ |

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SEADR*mn* [17:16] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEADR*mn*[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.35 LR*m*SEDADR*n*_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset is read. |
| 17 to 0 | SEADR*mn*[17:0] (*m* = 1 to 8) (*n* = 0 to 3) | 1-bit error detection address bank *n* |

**NOTE**

P1M-C, P1H-CE (P1M-C mode):

- LRAM_Base_Address = 0xFEBE_0000
- Error address SED : ( LRAM_Base_Address[31:18], SEADRmn[17:4], Bank_n[3:0])

P1H-C (4MB), P1H-C (8MB), P1H-CE (P1H-C (4MB) mode), P1H-CE (P1H-C (8MB) mode):

- LRAM_Base_Address = 0xFEBF_0000
- Error address SED : ( LRAM_Base_Address[31:18], SEADRmn[17:4], Bank_n[3:0])

For logical address of lower 4 bit of each bank, please replace as follows.

Bank0: SEADRm0[3:0] = Bank_0[3:0] = $0000_B$

Bank1: SEADRm1[3:0] = Bank_1[3:0] = $0100_B$

Bank2: SEADRm2[3:0] = Bank_2[3:0] = $1000_B$

Bank3: SEADRm3[3:0] = Bank_3[3:0] = $1100_B$

### (9) LRDEDADR*n*_PE1/PE2 — Local-RAM DED Address Register *n* (*n* = 0 to 3)

LRDEDADR*n*_PE1/PE2 (*n* = 0 to 3) register is used to hold the address when an error is detected. Error address capture trigger is same as corresponding DEDFn (*n* = 0 to 3) set in LRDERSTR_PE1/PE2. LRDEDADR*n*_PE1/PE2 (*n* = 0 to 3) is cleared by system reset or DSTCLR*n* (*n* = 0 to 3) in LRDERSTCLR_PE1/PE2 register.

**Access:** LRDEDADR*n*_PE1/PE2 can be read in 32-bit units.

**Address:**
LRDEDADR0_PE1: FFC6 54E0$_H$ LRDEDADR1_PE1: FFC6 54E4$_H$
LRDEDADR2_PE1: FFC6 54E8$_H$ LRDEDADR3_PE1: FFC6 54EC$_H$
LRDEDADR0_PE2: FFC6 58E0$_H$ LRDEDADR1_PE2: FFC6 58E4$_H$
LRDEDADR2_PE2: FFC6 58E8$_H$ LRDEDADR3_PE2: FFC6 58EC$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEADER*n* [17:16] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEADER*n*[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.36 LRDEDADR*n*_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset is read. |
| 17 to 0 | DEADR*n*[17:0] (*n* = 0 to 3) | 2-bit error detection address bank *n* |

**NOTE**

P1M-C, P1H-CE (P1M-C mode):

- LRAM_Base_Address = 0xFEBE_0000
- Error address DED : ( LRAM_Base_Address[31:18], DEADRmn[17:4], Bank_n[3:0])

P1H-C (4MB), P1H-C (8MB), P1H-CE (P1H-C (4MB) mode), P1H-CE (P1H-C (8MB) mode):

- LRAM_Base_Address = 0xFEBF_0000
- Error address DED : ( LRAM_Base_Address[31:18], DEADRmn[17:4], Bank_n[3:0])

For logical address of lower 4 bit of each bank, please replace as follows.

Bank0: DEADRm0[3:0] = Bank_0[3:0] = 0000$_B$
Bank1: DEADRm1[3:0] = Bank_1[3:0] = 0100$_B$
Bank2: DEADRm2[3:0] = Bank_2[3:0] = 1000$_B$
Bank3: DEADRm3[3:0] = Bank_3[3:0] = 1100$_B$

### (10) LRTSTCTL_PE1/PE2 — Local-RAM Test Control Register

LRTSTCTL_PE1/PE2 register control the ECC error injection. Writing this registers must be executed with PROT[1:0] = $01_B$.

**Access:** LRTSTCTL_PE1/PE2 can be read/written in 32-bit units.

**Address:** LRTSTCTL_PE1: FFC6 $5604_H$
LRTSTCTL_PE2: FFC6 $5A04_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT[1:0] | | — | — | — | — | — | — | — | — | — | — | — | — | ECCTST | DATSEL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.37 LRTSTCTL_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | ECCTST | ECC Test Mode Bit<br>0: Normal Mode<br>1: ECC Test Mode |
| 0 | DATSEL | Data Select Bits<br>Need ECCTST = 1<br>0: RAM Data Select<br>1: ECC bit Select |

**Table 28.38 Local RAM Test Mode**

| ECCTST | DATSEL | Write Data |
|---|---|---|
| 0 | 0 | Data and ECC bits is updated (normal mode) |
| 0 | 1 | Data and ECC bits is updated (normal mode) |
| 1 | 0 | Data only is updated. ECC bits is not updated |
| 1 | 1 | Lower 7 bits of data are written as ECC bits. Data is not updated. |

### (11) LRTDATBFn_PE1/PE2 — Local-RAM Test Data Read Buffer*n* (n = 0, 1)

**Access:** LRTDATBF0_PE1/PE2 can be read in 32-bit units.

**Address:** LRTDATBF0_PE1: FFC6 5608$_H$ LRTDATBF1_PE1: FFC6 560C$_H$
LRTDATBF0_PE2: FFC6 5A08$_H$ LRTDATBF1_PE2: FFC6 5A0C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | | — | LRDATABF[22:16] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | LRDATABF[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.39 LRTDATBF0_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | When read, the value after reset is read. |
| 22 to 16 | LRDATABF [22:16] | ECC Data at Word (2n + 1) are stored into this register in the read operation from LRAM when ECCTST = 1. |
| 15 to 7 | Reserved | When read, the value after reset is read. |
| 6 to 0 | LRDATABF[6:0] | ECC Data at Word (2n) are stored into this register in the read operation from LRAM when ECCTST = 1. |

#### 28.2.4.4 Test Function

Through appropriate register setting, desired values can be written as RAM data and to the ECC bits. Also, data in the RAM and the ECC bits can all be read.

**(1) Writing RAM data**

(a) Set the ECCTST bit in the local RAM test control register to 1 to set test mode.

(b) Set the corresponding DATSEL bit in the local RAM test control register to 0 to select RAM data for access when writing.

(c) When data is written to the local RAM, only the RAM data can be modified without updating the ECC bits.

How to exit this test mode:

(a) Set the ECCTST bit in the local RAM test control register to 0 to disable test mode (normal mode).

**(2) Reading RAM data**

(a) Set the ECCDIS bit in the local RAM ECC control register to 1 to disable ECC error detection and correction.

(b) Read the local RAM. Since neither error detection nor correction proceeds when the local RAM is read, the RAM data is read unchanged.

How to exit this test mode:

(a) Set the ECCDIS bit in the local RAM ECC control register to 0 to enable ECC error detection and correction.

**(3) Writing to the ECC bits**

(a) Set the ECCTST bit in the local RAM test control register to 1 to set test mode.

(b) Set the corresponding DATSEL bit in the local RAM test control register to 1 to select the ECC bits for access when writing.

(c) When data is written to the local RAM, only the ECC bits can be modified without updating the RAM data. At that time, bit[7:0] are respectively written to the ECC bits.

How to exit this test mode:

(a) Set the ECCTST bit in the local RAM test control register to 0 to disable test mode (normal mode).

### (4) Reading the ECC bits

(a) Set the ECCTST bit in the local RAM test control register to 1 to set test mode.

(b) When the local RAM is read, the ECC bits are stored in the bank corresponding to local RAM test data read buffer 0 or local RAM test data read buffer 1.

How to exit this test mode:

(a) Set the ECCTST bit in the local RAM test control register to 0 to disable test mode (normal mode).

### (5) Self-diagnosis of the ECC check function

Desired values can be written to the RAM data and/or ECC bits by following the procedure described in (1) or (3) above. Therefore, a fault can be injected by, for example, inverting appropriate bits of the RAM data and/or ECC bits. After that, self-diagnosis of the ECC decoder is possible by reading the local RAM in normal mode and checking the result of error correction or detection.

## 28.2.5 Global RAM ECC

### 28.2.5.1 Overview

The global RAM ECC is summarized in the table below.

The below described ECC protection consider all possible global RAM accesses from PE1, PE2, H-Bus, System Interconnect and read access for read-modify-write processing unit of 8-bit and 16-bit data.

Further the related Suffix will be used:

“_GRAMC”: PE1, PE2, read-accesses for read-modify-write processing when 8-bit or 16-bit data

“_VCI2GRAM”: system interconnect

“_AXI2GRAM” : H-Bus

**Table 28.40 Global RAM ECC**

| Item | Description |
|---|---|
| ECC error detection and correction | ECC error detection and correction can be either enabled or disabled.<br>When enabled, either of the following settings can be selected.<br>• ECC error detection and correction are carried out (2-bit error detection and 1-bit error detection and correction are carried out).<br>• ECC error detection is carried out (2-bit error detection and 1-bit error detection are carried out).<br>When disabled, neither error detection nor correction is carried out.<br>In the initial state, the ECC function is enabled, and 1-bit error detection and correction and 2-bit error detection and notification are carried out. |
| Address ECC | Address ECC check can be either enabled or disabled.<br>Address ECC detects address error on bus between before Global RAM and master.<br>In the initial state, 1-bit error detection, 2-bit error detection and notification are carried out.<br>**NOTE**<br>Even though ECC circuit with 1-bit error correction capability is implemented, still a 1-bit error indicates a sever fault and shall be properly treated in accordance with the safety requirements (see **Section 28.2.9.1**) |
| Error notification | Upon occurrence of an ECC error or Address ECC error, it is notified to the Error Control Module.<br><br>**ECC Error:**<br>• Error notification can be either enabled or disabled upon detection of an ECC 2-bit error.<br>• Error notification can be either enabled or disabled upon detection of an ECC 1-bit error.<br>In the initial state, error notification is enabled upon detection of an ECC 2-bit error, and error notification is enabled upon detection of an ECC 1-bit error.<br><br>**Address ECC Error:**<br>• Error notification can be either enabled or disabled upon detection of an ECC 2-bit error.<br>• Error notification can be either enabled or disabled upon detection of an ECC 1bit error.<br>In the initial state, error notification is enabled upon detection of an ECC 2-bit and ECC 1-bit error.<br><br>**Overflow Error:**<br>• Error notification can be either enable or disable upon detection of an address buffer overflow error for Data ECC 1-bit error.<br>In the initial state, error notification is enabled upon detection of an address buffer overflow error.<br>ECC 2-bit error, address ECC 2-bit error, ECC 1-bit error, overflow error and address ECC 1-bit error are handled as each individual source. An ECC 1-bit error signal is only issued to the ECM, if the ECC 1-bit error address is not yet stored in the error address buffer. |
| Error status | A status register is provided, which indicates the statuses of ECC 2-bit error detection, ECC 1-bit error detection, and Address ECC error detection. If an error occurs while no error status is set, the corresponding status is set.<br>The error status can be cleared using the clear register. |
| Address capture | Multi-stage address buffers are provided for a 1-bit data error, and check is performed in each stage. The address is latched when the state of each stage is clear.<br>• 1-bit data error: 32 stages<br>• 2-bit data and address ECC, shared: One stage<br>The address is captured when an ECC 2-bit error, an ECC 1-bit error, or an Address ECC error is detected.<br>The error status serves as the enable bit of the capture address. |

### 28.2.5.2 List of Registers

Table 28.41 List of Registers (1/3)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC64000$_H$ | UGRERRINT | Global-RAM Error Information Control Register | R/W | 0000 0073$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC64004$_H$ | UGRSERSTCLR | Global-RAM ECC SED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC64008$_H$ | UGRDERSTCLR | Global-RAM ECC DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6400C$_H$ | UGROVFSTR | Global-RAM Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC64020$_H$ | UGRSERSTR | Global-RAM ECC SED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC64030$_H$ | UGRDERSTR | Global-RAM ECC DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC64040$_H$ | UGR1SEDADR | Global-RAM 1 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64044$_H$ | UGR2SEDADR | Global-RAM 2 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64048$_H$ | UGR3SEDADR | Global-RAM 3 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6404C$_H$ | UGR4SEDADR | Global-RAM 4 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64050$_H$ | UGR5SEDADR | Global-RAM 5 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64054$_H$ | UGR6SEDADR | Global-RAM 6 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64058$_H$ | UGR7SEDADR | Global-RAM 7 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6405C$_H$ | UGR8SEDADR | Global-RAM 8 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64060$_H$ | UGR9SEDADR | Global-RAM 9 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64064$_H$ | UGR10SEDADR | Global-RAM 10 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64068$_H$ | UGR11SEDADR | Global-RAM 11 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6406C$_H$ | UGR12SEDADR | Global-RAM 12 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64070$_H$ | UGR13SEDADR | Global-RAM 13 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64074$_H$ | UGR14SEDADR | Global-RAM 14 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64078$_H$ | UGR15SEDADR | Global-RAM 15 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6407C$_H$ | UGR16SEDADR | Global-RAM 16 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64080$_H$ | UGR17SEDADR | Global-RAM 17 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64084$_H$ | UGR18SEDADR | Global-RAM 18 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |

Table 28.41 List of Registers (2/3)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC64088$_H$ | UGR19SEDADR | Global-RAM 19 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6408C$_H$ | UGR20SEDADR | Global-RAM 20 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64090$_H$ | UGR21SEDADR | Global-RAM 21 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64094$_H$ | UGR22SEDADR | Global-RAM 22 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64098$_H$ | UGR23SEDADR | Global-RAM 23 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6409C$_H$ | UGR24SEDADR | Global-RAM 24 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC640A0$_H$ | UGR25SEDADR | Global-RAM 25 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC640A4$_H$ | UGR26SEDADR | Global-RAM 26 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC640A8$_H$ | UGR27SEDADR | Global-RAM 27 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC640AC$_H$ | UGR28SEDADR | Global-RAM 28 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC640B0$_H$ | UGR29SEDADR | Global-RAM 29 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC640B4$_H$ | UGR30SEDADR | Global-RAM 30 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC640B8$_H$ | UGR31SEDADR | Global-RAM 31 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC640BC$_H$ | UGR32SEDADR | Global-RAM 32 ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC640C0$_H$ | UGR00DEDADR | Global-RAM ECC DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC64100$_H$ | GRECCCTL_GRAMC | Global-RAM ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC64104$_H$ | GRTSTCTL | Global-RAM Test Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC64108$_H$ | GRTDATBF0L | Global-RAM Test Data Read Buffer (Lower 32 bit of Bank0) | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6410C$_H$ | GRTDATBF0H | Global-RAM Test Data Read Buffer (Upper 32 bit of Bank0) | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC64110$_H$ | GRTDATBF1L | Global-RAM Test Data Read Buffer (Lower 32 bit of Bank 1) | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC64114$_H$ | GRTDATBF1H | Global-RAM Test Data Read Buffer (Upper 32 bit of Bank 1) | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC64118$_H$ | GRDECINBF0 | GRAMC ECC Decoder Input Data Buffer 0 | R/W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6411C$_H$ | GRDECINBF1 | GRAMC ECC Decoder Input Data Buffer 1 | R/W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC64200$_H$ | GRECCCTL_VCI2GRAM | Global-RAM ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |

Table 28.41 List of Registers (3/3)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC64400$_H$ | GRECCCTL_PE1*1 | Global-RAM ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP1 | — |
| FFC64600$_H$ | GRECCCTL_PE2*1,*2 | Global-RAM ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP2 | — |
| FFC64E00$_H$ | GRECCCTL_AXI2GRAM | Global-RAM ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |

Note 1. This register doesn't have any functions.

Note 2. P1H-CE, P1H-C devices only. P1M-C device doesn't have these registers.

**Note:** The registers with symbols "_GRAMC", "_VCI2GRAM", and "_AXI2GRAM" as suffixes are provided to the particular ECC controllers: the registers with "_GRAMC" are provided to the ECC controller for access from the CPU1 and CPU2 to GRAM, and for read-accesses for read-modify-write processing when 8-bit or 16-bit data is written to GRAM, the registers with "_VCI2GRAM" are provided to the ECC controller for access from the system interconnect to GRAM, and the registers with "_AXI2GRAM" are provided to the ECC controller for access from H-bus.

### 28.2.5.3 Details of Registers

#### (1) UGRERRINT — Global-RAM Error Information Control Register

UGRERRINT register controls whether error information is reported to ECM, when address/data ECC 2-bit error, address/data ECC 1-bit error, and ECC 1-bit error overflow are detected.

**Access:** UGRERRINT can be read/written in 32/16/8-bit units.

**Address:** FFC6 $4000_H$

**Value after reset:** 0000 $0073_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | SEOVFIE | ADEDIE | ASEDIE | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R | R/W | R/W |

**Table 28.42 UGRERRINT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 | SEOVFIE | ECC 1-bit error overflow report enable bit<br>Overflow report control bit when 1-bit error overflow flag (SERROVF) in UGROVFSTR register is set.<br>0: ECC 1-bit error overflow report disabled<br>1: ECC 1-bit error overflow report enabled |
| 5 | ADEDIE | Address ECC 2-bit error on bus report enable bit<br>Control error report of 2-bit error detection when address ECC error detection/correction is enabled.<br>0: Address ECC 2-bit error report disabled<br>1: Address ECC 2-bit error report enabled |
| 4 | ASEDIE | Address ECC 1-bit error on bus report enable bit<br>Control error report of 1-bit error detection when address ECC error detection/correction is enabled.<br>0: Address ECC 1-bit error report disabled<br>1: Address ECC 1-bit error report enabled |
| 3, 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when ECC error detection/correction is enabled<br>0: ECC 2-bit error report disabled<br>1: ECC 2-bit error report enabled |
| 0 | SEDIE | ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when ECC error detection/correction is enabled<br>0: ECC 1-bit error report disabled<br>1: ECC 1-bit error report enabled |

### (2) UGRSERSTCLR — Global-RAM ECC SED Status Clear Register

UGRSERSTCLR register is used to clear SEDF[0:31] in UGRSERSTR, SERROVF in UGROVFSTR, and error address in UGR[1:32]SEDADR. This is write only register and read value is always “0”. UGRSERSTCLR register has a lower priority than a set factor. Priority is given to a set factor when UGRSERSTCLR and a set factor compete. A set factor means a trigger of setting SEDF[0:31] in UGRSERSTR, setting SERROVF in UGROVFSTR, or capturing an error address in UGR[1:32]SEDADR.

**Access:** UGRSERSTCLR can be written in 32/16/8-bit units.

**Address:** FFC6 $4004_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSTCLR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSTCLR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 28.43 UGRSERSTCLR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SSTCLR[31:0] | Data ECC 1-bit error flag clear n th<br>1: Following flag and address clear<br>UGRSERSTR.SEDFn,<br>UGROVFSTR.SERROVF,<br>UGRnSEDADR.SEADR |

### (3) UGRDERSTCLR — Global-RAM ECC DED Status Clear Register

UGRDERSTCLR register is used to clear ADEDF/ASEDF and DEDF in UGRDERSTR and error address in UGR00DEDADR. This is write only register and read value is always “0”.
UGRDERSTCLR register has a lower priority than a set factor. Priority is given to a set factor when UGRDERSTCLR and a set factor compete. A set factor means a trigger of setting ADEDF/ASEDF and DEDF in UGRDERSTR.

**Access:** UGRDERSTCLR can be written in 32/16/8-bit units.

**Address:** FFC6 4008$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.44 UGRDERSTCLR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | DSTCLR | Data ECC 2-bit error flag and address ECC error flag clear<br>1: Following flag and address clear<br>UGRDERSTR.ADEDF,<br>UGRDERSTR.ASEDF,<br>UGRDERSTR.DEDF,<br>UGR00DEDADR.DEADR |

### (4) UGROVFSTR — Global-RAM Error Count Overflow Status Register

UGROVFSTR register monitor if error overflow occurs. Overflow occurs when a SED with the error address never captured is detected in the case that ECC 1-bit error status register is full. If the ECC 1-bit error status register is full and the error address is the same as one of the error addresses already captured, this flag is not set. SERROVF flag is cleared by system reset, or cleared when at least one of SSTCLR[0:31] in UGRSERSTCLR register is asserted.

**Access:** UGROVFSTR can be read in 32/16/8-bit units.

**Address:** FFC6 400C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.45 UGROVFSTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SERROVF | 1-bit error overflow flag<br>This error flag is set if the followings occur.<br>• A SED occurrence when all bits(SEDF [0:31]) of ECC 1-bit error status register (UGRSERSTR) are set.<br>• If the ECC 1-bit error status register is full and the error address is the same as one of the error addresses already captured, this flag is not set. |

**Note:** As the overflow flag, only SED flags are prepared. Because DED is prepared for only one error address buffer, an overflow flag for DED is not necessary.

### (5) UGRSERSTR — Global-RAM ECC SED Status Register

UGRSERSTR is the error monitor register. Each error flag is “0”and when a new error occurs, an error status flag is set. An error flag is set at the lowest number empty bit of UGRSERSTR (e.g. If SEDF[0][1][3] have been set to “1” and all other bits have been empty, a next flag is set to SEDF[2].). If multiple SED causes are detected simultaneously and there are empty bits sufficiently, detected SEDs are all set (e.g. if a SED and another SED which occurs at a different address are detected simultaneously, both SEDFs are set.). However, if SEDs occur at a same address simultaneously, only one of errors is set according to the fixed priority. The priority order is PE1, PE2, system interconnect and H-bus. (e.g. if a SED which is input from PE1 path and an SED which is input from PE2 path are detected simultaneously and those occur at the same address, only SEDF from the PE1 path is set and SEDF from the PE2 path is not set.) And also, an error address which has already captured in UGR[1:32]SEDADR must not be captured again. UGRSERSTR register is cleared by system reset or SSTCLR[0:31] in UGRSERSTCLR register.

**Access:** UGRSERSTR can be read in 32/16/8-bit units.

**Address:** FFC6 4020$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEDF[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEDF[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.46 UGRSERSTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SEDF[31:0] | ECC 1-bit error monitor flag SEDF[m] in m$^{th}$ buffer<br>Condition to “0”: system reset or write “1” to SSTCLRn in UGRSERSTCLR<br>Condition to “1”: ECC 1-bit error is detected. |

### (6) UGRDERSTR — Global-RAM ECC DED Status Register

UGRDERSTR is the error monitor register. All error flags are “0” and when a new error occurs, an error status flag is set. If multiple error causes which are input from each different slave port are detected simultaneously, only one of detected errors is set according to the fixed priority. The reason is that there is only one 2-bit error Address Register. The priority order is PE1, PE2, system interconnect and H-bus. (e.g. if a DED which is input from PE1 path and an ASED which is input from PE2 path are detected simultaneously, only DEDF from the PE1 path is set and ASEDF from the PE2 path is not set.) If multiple error causes occur from same slave port simultaneously, detected errors are all set. (e.g. if DED and ASED are simultaneously input from PE1 path, both DEDF and ASEDF are set.) UGRDERSTR register is cleared by system reset or DSTCLR in UGRDERSTCLR register.

**Access:** UGRDERSTR can be read in 32/16/8-bit units.

**Address:** FFC6 $4030_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ADEDF | ASEDF | — | — | — | DEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.47 UGRDERSTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. |
| 5 | ADEDF | Address ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to DSTCLR in UGRDERSTCLR<br>Condition to “1”: ADEDF is “0” and address ECC 2-bit error is detected. |
| 4 | ASEDF | Address ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to DSTCLR in UGRDERSTCLR<br>Condition to “1”: ASEDF is “0” and address ECC 1-bit error is detected. |
| 3 to 1 | Reserved | When read, the value after reset is read. |
| 0 | DEDF | ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to DSTCLR in UGRDERCTCLR<br>Condition to “1”: DEDF is “0” and ECC 2-bit error is detected. |

**Note:** As the ASEDF flag, this register has been named for capturing ECC 2-bit error (DED) error status, but the register captures Address ECC 1-bit error (ASED) too. Because ASED is also an emergency error and S/W should check the status immediately, ASED is treated like a DED error.

### (7) UGR*n*SEDADR — Global-RAM *n* ECC SED Address Register (*n* = 1 to 32)

UGR*n*SEDADR register is used to hold the address when a 1-bit error is detected. Error address capture trigger is same as corresponding SEDF(*n*-1) set in UGRSERSTR. UGR*n*SEDADR is cleared by system reset or SSTCLR(*n*-1) in UGRSERSTCLR register.

**Access:** UGR*n*SEDADR can be read in 32-bit units.

**Address:**
UGR1SEDADR: FFC6 4040$_H$
UGR2SEDADR: FFC6 4044$_H$
UGR3SEDADR: FFC6 4048$_H$
UGR4SEDADR: FFC6 404C$_H$
UGR5SEDADR: FFC6 4050$_H$
UGR6SEDADR: FFC6 4054$_H$
UGR7SEDADR: FFC6 4058$_H$
UGR8SEDADR: FFC6 405C$_H$
UGR9SEDADR: FFC6 4060$_H$
UGR10SEDADR: FFC6 4064$_H$
UGR11SEDADR: FFC6 4068$_H$
UGR12SEDADR: FFC6 406C$_H$
UGR13SEDADR: FFC6 4070$_H$
UGR14SEDADR: FFC6 4074$_H$
UGR15SEDADR: FFC6 4078$_H$
UGR16SEDADR: FFC6 407C$_H$
UGR17SEDADR: FFC6 4080$_H$
UGR18SEDADR: FFC6 4084$_H$
UGR19SEDADR: FFC6 4088$_H$
UGR20SEDADR: FFC6 408C$_H$
UGR21SEDADR: FFC6 4090$_H$
UGR22SEDADR: FFC6 4094$_H$
UGR23SEDADR: FFC6 4098$_H$
UGR24SEDADR: FFC6 409C$_H$
UGR25SEDADR: FFC6 40A0$_H$
UGR26SEDADR: FFC6 40A4$_H$
UGR27SEDADR: FFC6 40A8$_H$
UGR28SEDADR: FFC6 40AC$_H$
UGR29SEDADR: FFC6 40B0$_H$
UGR30SEDADR: FFC6 40B4$_H$
UGR31SEDADR: FFC6 40B8$_H$
UGR32SEDADR: FFC6 40BC$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | SEADR[20:16] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.48 UGR*n*SEDADR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. |
| 20 to 0 | SEADR[20:0] | 1-bit error detection address |

**NOTE**

P1M-C:

- GRAM_Base_Address = 0xFEED_8000
- Error address SED : ( GRAM_Base_Address[31:21] , SEDARn[20:0] )

P1H-C (4MB), P1H-C (8MB):

- GRAM_Base_Address = 0xFEE8_8000
- Error address SED : ( GRAM_Base_Address[31:21] , SEDARn[20:0] )

P1H-CE (P1M-C mode), P1H-CE (P1H-C (4MB) mode), P1H-CE (P1H-C (8MB) mode):

- GRAM_Base_Address = 0xFEE5_8000
- Error address SED : ( GRAM_Base_Address[31:21] , SEDARn[20:0] )

### (8) UGR00DEDADR — Global-RAM ECC DED Address Register

UGR00DEDADR register is used to hold the address when an error is detected. Error address capture trigger is same as corresponding ADEDF/ASEDF/DEDF set in UGRDERSTR. If this register has captured an error address, the register doesn't capture any more address. UGR00DEDADR is cleared by system reset or DSTCLR in UGRDERSTCLR register.

**Access:** UGR00DEDADR can be read in 32-bit units.

**Address:** FFC6 40C0$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | DEADR[20:16] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.49 UGR00DEDADR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. |
| 20 to 0 | DEADR[20:0] | Data 2-bit error or address 1 bit or address 2-bit error detection address |

**NOTE**

P1M-C:

- GRAM_Base_Address = 0xFEED_8000
- Error address DED : ( GRAM_Base_Address[31:21] , DEADR[20:0] )

P1H-C (4MB), P1H-C (8MB):

- GRAM_Base_Address = 0xFEE8_8000
- Error address DED : ( GRAM_Base_Address[31:21] , DEADR[20:0] )

P1H-CE (P1M-C mode), P1H-CE (P1H-C (4MB) mode), P1H-CE (P1H-C (8MB) mode):

- GRAM_Base_Address = 0xFEE5_8000
- Error address DED : ( GRAM_Base_Address[31:21] , DEADR[20:0] )

### (9) GRECCCTL_GRAMC — Global-RAM ECC Control Register

GRECCCTL_GRAMC register controls the ECC error detection/correction and 1-bit error correction on address and data ECC. Writing ECC control register must be executed with PROT[1:0] = $01_B$.

**Access:** GRECCCTL_GRAMC can be read/written in 32/16-bit units.

**Address:** FFC6 $4100_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT[1:0] | | — | — | — | — | — | — | — | — | — | ASECDIS | AECCDIS | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W |

**Table 28.50 GRECCCTL_GRAMC register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | ASECDIS | Address ECC 1-bit error correction enable bit<br>When using ECC error detection/correction (AECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is carried out when 1-bit error is detected<br>1: Correction is not carried out when 1-bit error is detected |
| 3 | AECCDIS | Address ECC disable bit<br>Setting ECC error detection/correction to enable/disable.<br>0: ECC error detection/correction is enable<br>1: ECC error detection/correction is disable |
| 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SECDIS | Data ECC 1-bit error correction enable bit<br>When using ECC error detection/correction (ECCDIS = 0), setting 1-bit error correction to<br>enable/disable.<br>0: Correction is carried out when 1-bit error is detected<br>1: Correction is not carried out when 1-bit error is detected |
| 0 | ECCDIS | Data ECC disable bit<br>Setting ECC error detection/correction to enable/disable.<br>0: ECC error detection/correction is enable<br>1: ECC error detection/correction is disable |

### (10) GRTSTCTL — Global-RAM Test Control Register

GRTSTCTL register control the ECC error injection. Writing this registers must be executed with PROT[1:0] = $01_B$.

**Access:** GRTSTCTL can be read/written in 32/16-bit units.

**Address:** FFC6 $4104_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT[1:0] | | — | — | — | — | — | — | — | — | — | — | ECCTST | DECINEN | DATSEL[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 28.51 GRTSTCTL register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 | ECCTST | ECC Test Mode Bit<br>0: Normal Mode<br>1: ECC Test Mode |
| 2 | DECINEN | GRAMC ECC Decoder Error Injection Enable Bit<br>Need ECCTST = 1.<br>0: Not Select Data of ECC Decoder Input Buffer<br>1: Select Data of ECC Decoder Input Buffer |

**Table 28.51 GRTSTCTL register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1, 0 | DATSEL[1:0] | Read Buffer Storage Data Select 0 and 1<br>This bit is valid when ECCTST = 1. This bit selects the value to be stored in the read buffer GRTDATBFn, the value to be written to each field, and the value to be input to the ECC decoder for use in updating of data at the time of RMW access.<br>Write a value to this bit simultaneously with setting (PROT1, PROT0) = (0, 1).<br>00:<br>– GRTDATBFn: When reading involves an RMW cycle, the ECC bits are stored.<br>– Global RAM: When writing involves an RMW cycle, the data area that is updated depends on the unit of access and the location accessed. The ECC bits are not updated.<br>01:<br>– GRTDATBFn: When reading involves an RMW cycle, the ECC bits are stored.<br>– Global RAM: When writing involves an RMW cycle, only the ECC bits are updated. The data area is not updated.<br>10:<br>– GRTDATBFn: When access is RMW, this register holds the result of ECC decoding for the data read out in the read portion of the RMW cycle. Its value is not updated in the case of access that is not RMW.<br>– Global RAM: Operates in the same way as in normal operating mode (i.e. when ECCTST = 0).<br>11:<br>– GRTDATBFn: When access is RMW, this register holds the result of ECC decoding for use in the updating of data at the time of RMW access. Its value is not updated in the case of access that is not RMW.<br>– Global RAM: Operates in the same way as in normal operating mode (i.e. when ECCTST = 0).<br>– When and only when these settings are the case, the value of the GRDECINBF0 or 1 register is input to the ECC decoder for use in updating of data at the time of RMW access instead of the write data being sent to the CPU etc.<br>In any case, the result of reading by the CPU, DMAC, etc. is the same value as would normally be read out. |

### (11) GRTDATBF[0/1][L/H] — Global-RAM Test Data Read Buffer (Lower/Upper 32 bit of Bank0/1)

**Access:** GRTDATBF can be read in 32/16/8-bit units.

**Address:** GRTDATBF0L: FFC6 4108$_H$
GRTDATBF0H: FFC6 410C$_H$
GRTDATBF1L: FFC6 4110$_H$
GRTDATBF1H: FFC6 4114$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GRTDATBF[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GRTDATBF[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.52 GRTDATBF register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | GRTDATBF [31:0] | These bits are valid when while ECCTST = 1 (selecting test mode) in the global RAM test control register.<br>• When (DATSEL1, DATSEL0) = (0, 0) or (0, 1)<br>When reading from the RAM, the ECC bits are respectively stored in GRTDATBF[6:0]. 0 is stored in GRTDATBF[31:7].<br>• When (DATSEL1, DATSEL0) = (1, 0)<br>When access is RMW, the output data from the ECC decoder for reading (after updating) are stored in GRTDATBF[31:0].<br>• When (DATSEL1, DATSEL0) = (1, 1)<br>When access is RMW, the output data from the ECC decoder for use in updating of data (after updating) are stored in GRTDATBF[31:0]. |

**NOTE**

The GRAM consists of two banks, where bank 0 is synonymous to bank A and bank 1 is synonymous to bank B for this register.

The GRAM interface has 64 bit width, but the ECC is always made on both 32 bit words. Therefore the ECC/data of the 64 bit aligned lower 32 bits can be found in the GRTDATBFnL register, while the ECC/data of the 64 bit aligned upper 32 bits is found in the GRTDATBFnH register.

Please note that the location of the ECC/data of a 32 bit variable (with 32 bit alignment) depends on the address bit A2 (whereby A1 and A0 are 0). When A2 is 0, the ECC/data can be found in GRTDATBFnL register, and when A2 is 1 the ECC/data can be found in GRTDATBFnH register.

### (12) GRDECINBF0 — GRAMC ECC Decoder Input Data Buffer 0

**Access:** GRDECINBF0 can be read/written in 32/16/8-bit units.

**Address:** FFC6 4118$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GRDECINBF0[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GRDECINBF0[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.53 GRDECINBF0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | GRDECINBF0 [31:0] | These bits are valid when ECCTST = 1 (selecting test mode) in the global RAM test control register.<br>When DECINEN = 1, the value of this register is input to the ECC decoder as data for use in updating in response to the execution of an RMW instruction. The value is treated as if it were 32 bits of data from RAM.<br>This register is common to banks A and B and provides the values for both the 32 higher-order bits and the 32 lower-order bits. |

### (13) GRDECINBF1 — GRAMC ECC Decoder Input Data Buffer 1

**Access:** GRDECINBF1 can be read/written in 32/16/8-bit units.

**Address:** FFC6 411C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | GRDECINBF1[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.54 GRDECINBF1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 to 0 | GRDECINBF1 [6:0] | These bits are valid when ECCTST = 1 (selecting test mode) in the global RAM test control register.<br>When DECINEN = 1, the value of this register is input to the ECC decoder as data for use in updating in response to the execution of an RMW instruction. The value is treated as if it were 7 bits of data from the ECC.<br>This register is common to banks A and B and provides the values for both the 32 higher-order bits and the 32 lower-order bits. |

### (14) GRECCCTL_VCI2GRAM/PE1/PE2/AXI2GRAM — Global-RAM ECC Control Register

GRECCCTL_VCI2GRAM/PE1/PE2/AXI2GRAM register control the ECC error detection/correction and 1-bit error correction. Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

For the related access part, please consider this register:

GRECCCTL_VCI2GRAM: access over system interconnect

GRECCCTL_PE1: *1

GRECCCTL_PE2: *1

GRECCCTL_AXI2GRAM: access over H-BUS.

**Note 1.** GRECCCTL_PE1/PE2 registers can be written and read. However even if these registers are set, the setting doesn't affect the operation.
Use GRECCCTL_GRAMC to disable/enable GRAM ECC decoders of PE1 and PE2.

**Access:** GRECCCTL_VCI2GRAM/PE1/PE2/AXI2GRAM can be read/written in 32/16-bit units.

**Address:**

| | |
|---|---|
| GRECCCTL_VCI2GRAM: | FFC6 $4200_H$ |
| GRECCCTL_PE1: | FFC6 $4400_H$ |
| GRECCCTL_PE2: | FFC6 $4600_H$ |
| GRECCCTL_AXI2GRAM: | FFC6 $4E00_H$ |

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT[1:0] | | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.55 GRECCCTL_VCI2GRAM/PE1/PE2/AXI2GRAM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SECDIS | Data ECC 1-bit error correction enable bit<br>When using ECC error detection/correction (ECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is carried out when 1-bit is detected<br>1: Correction is not carried out when 1-bit is detected |
| 0 | ECCDIS | Data ECC disable bit<br>Setting ECC error detection/correction to enable/disable.<br>0: ECC error detection/correction is enable<br>1: ECC error detection/correction is disable |

#### 28.2.5.4 Test Function

Through appropriate register setting, desired values can be written as RAM data and to the ECC bits. Also, data in the RAM and the ECC bits and the ECC decoder output data for RMW can all be read.

It is possible to input the desired data in the ECC decoder for RMW.

**(1) Writing RAM data**

(a) Set the ECCTST bit in the global RAM test control register to 1 to set test mode.

(b) Set the corresponding DATSEL1 to 0 and DATSEL0 to 0 in the global RAM test control register to select RAM data for access when writing.

(c) When data is written to the global RAM, only the RAM data can be modified without updating the ECC bits.

How to exit this test mode:

(a) Set the ECCTST bit in the global RAM test control register to 0 to disable test mode (normal mode).

**(2) Reading RAM data**

(a) Set the ECCDIS bit in the global RAM ECC control register to 1 to disable ECC error detection and correction.

(b) Read the global RAM. Since neither error detection nor correction proceeds when the global RAM is read, the RAM data is read unchanged.

How to exit this test mode:

(a) Set the ECCDIS bit in the global RAM ECC control register to 0 to enable ECC error detection and correction.

**(3) Writing to the ECC bits**

(a) Set the ECCTST bit in the global RAM test control register to 1 to set test mode.

(b) Set the corresponding DATSEL1 bit to 0 and DATSEL0 bit to 1 in the global RAM test control register to select the ECC bits for access when writing.

(c) When data is written to the global RAM, only the ECC bits can be modified without updating the RAM data. At that time, bit[7:0] are respectively written to the ECC bits.

How to exit this test mode:

(a) Set the ECCTST bit in the global RAM test control register to 0 to disable test mode (normal mode).

**(4) Reading the ECC bits**

(a) Set the ECCTST bit in the global RAM test control register to 1 to set test mode.

(b) Setting the DATSEL1 and DATSEL0 bits in the global RAM test control register to 0 and 1, respectively, to select the ECC for access when reading.

(c) When data in the global RAM is read, the ECC bits are stored in the corresponding register from among global RAM test data read buffers 0 to 3.

How to exit this test mode:

(a) Set the ECCTST bit in the global RAM test control register to 0 to disable test mode (normal mode).

### (5) Self-diagnosis of the ECC check function for the access ports

Desired values can be written to the RAM data and/or ECC bits by following the procedure described in (1) or (3) above. Therefore, a fault can be injected by, for example, inverting appropriate bits of the RAM data and/or ECC bits. After that, self-diagnosis of the ECC decoder is possible by reading the global RAM in normal mode and checking the result of error correction or detection.

### (6) Self-diagnosis of the ECC decoder for the data read out in an RMW operation

(a) Suitable erroneous values are injected as RAM data or to the ECC bits by following procedure (1) or (3) above.

(b) Setting the DATSEL1 and DATSEL0 bits in the test control registers for the global RAM to 1 and 0, respectively, makes the output data from the ECC decoder for the data read out in an RMW operation the target for reading.

(c) After RMW processing for the global RAM proceeds, the data read out in an RMW operation for global RAM are stored in the corresponding register from among global RAM test data read buffers 0 to 3. Checking the result allows self-diagnosis of the ECC decoder for the data read out in an RMW operation.

### (7) Self-diagnosis of the ECC decoder for the data being updated at the time of RMW access

(a) Setting the DATSEL1 and DATSEL0 bits in the global RAM test control register to 1 makes the output data from the ECC decoder for the data being updated at the time of RMW access the target for reading.

(b) Through the above setting, the input data from the ECC decoder for the data being updated at the time of RMW access is switched to ECC decoder input buffer 0 or 1 (GRDECINBF0 or 1) from write data sent from the access ports. As a result, suitable erroneous values can be injected by setting an appropriate value in ECC decoder input buffer 0 or 1.

## 28.2.6 Instruction Cache ECC and EDC

### 28.2.6.1 Overview

The instruction cache ECC is summarized in the table below.

**Table 28.56 Instruction Cache ECC**

| Item | Description |
|---|---|
| ECC error detection | ECC error detection can be either enabled or disabled.<br><br>When enabled, the following settings can be selected.<br>• ECC error detection is carried out (2-bit error detection and 1-bit error detection are carried out).<br><br>When disabled, error detection is not carried out.<br>In the initial state, the ECC function is enabled, and 1-bit error detection, and notification, and 2-bit error detection and notification are carried out. |
| Error notification | Upon occurrence of an ECC error, it is notified to the Error Control Module.<br><br>**ECC Error:**<br>• Error notification can be either enabled or disabled upon detection of an ECC 2-bit error.<br>• Error notification can be either enabled or disabled upon detection of an ECC 1-bit error.<br>In the initial state, notification of the ECC 2-bit error is enabled, and notification of the ECC 1-bit error is enabled.<br><br>An ECC 1-bit error signal is only issued to the ECM, if the ECC 1-bit error address is not yet stored in the error address buffer. |
| Error status | A status register is provided, which indicates the status of ECC 2-bit error detection and ECC 1-bit error detection. If an error occurs while no error status is set, the corresponding status is set.<br>The error status can be cleared using the clear register. |
| Address capture | If an ECC error occurs while no error status is set, the address at which the associated error has occurred is captured.<br>The address is captured when an ECC 2-bit error or an ECC 1-bit error is detected.<br>The error status serves as the enable bit of the capture address. |

### 28.2.6.2 List of Registers

Table 28.57 List of Registers (1/3)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 0400$_H$ | IDECCCTL_PE1 | Instruction Cache Data RAM Data ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP1 | — |
| FFC6 0404$_H$ | IDERRINT_PE1 | Instruction Cache Data RAM Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 0408$_H$ | IDSERSTCLR_PE1 | Instruction Cache Data RAM Data ECC SED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 040C$_H$ | IDDERSTCLR_PE1 | Instruction Cache Data RAM Data ECC DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 0410$_H$ | IDOVFSTR_PE1 | Instruction Cache Data RAM Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 0420$_H$ | IDSERSTR_PE1 | Instruction Cache Data RAM Data ECC SED Status Register (PE1) | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 0450$_H$ | IDDERSTR_PE1 | Instruction Cache Data RAM Data ECC DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 0460$_H$ | IDSEDADR0_PE1 | Instruction Cache Data RAM (Bank0) Data ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 0464$_H$ | IDSEDADR1_PE1 | Instruction Cache Data RAM (Bank1) Data ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 04E0$_H$ | IDDEDADR0_PE1 | Instruction Cache Data RAM (Bank0) Data ECC DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 04E4$_H$ | IDDEDADR1_PE1 | Instruction Cache Data RAM (Bank1) Data ECC DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 0600$_H$ | IDECCCTL_PE2[*1] | Instruction Cache Data RAM Data ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP2 | — |
| FFC6 0604$_H$ | IDERRINT_PE2[*1] | Instruction Cache Data RAM Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 0608$_H$ | IDSERSTCLR_PE2[*1] | Instruction Cache Data RAM Data ECC SED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 060C$_H$ | IDDERSTCLR_PE2[*1] | Instruction Cache Data RAM Data ECC DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 0610$_H$ | IDOVFSTR_PE2[*1] | Instruction Cache Data RAM Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 0620$_H$ | IDSERSTR_PE2[*1] | Instruction Cache Data RAM Data ECC SED Status Register (PE2) | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 0650$_H$ | IDDERSTR_PE2[*1] | Instruction Cache Data RAM Data ECC DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |

Table 28.57 List of Registers (2/3)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 0660$_H$ | IDSEDADR0_PE2*1 | Instruction Cache Data RAM (Bank0) Data ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 0664$_H$ | IDSEDADR1_PE2*1 | Instruction Cache Data RAM (Bank1) Data ECC SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 06E0$_H$ | IDDEDADR0_PE2*1 | Instruction Cache Data RAM (Bank0) Data ECC DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 06E4$_H$ | IDDEDADR1_PE2*1 | Instruction Cache Data RAM (Bank1) Data ECC DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 1400$_H$ | ITECCCTL_PE1 | Instruction Cache Tag RAM ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP1 | — |
| FFC6 1404$_H$ | ITERRINT_PE1 | Instruction Cache Tag RAM Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 1408$_H$ | ITSERSTCLR_PE1 | Instruction Cache Tag RAM ECC SED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 140C$_H$ | ITDERSTCLR_PE1 | Instruction Cache Tag RAM ECC DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 1410$_H$ | ITOVFSTR_PE1 | Instruction Cache Tag RAM Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 1420$_H$ | ITSERSTR_PE1 | Instruction Cache Tag RAM ECC SED Status Register (PE1) | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 1450$_H$ | ITDERSTR_PE1 | Instruction Cache Tag RAM ECC DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 1460$_H$ | ITSEDADR_PE1 | Instruction Cache Tag RAM SED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 14E0$_H$ | ITDEDADR_PE1 | Instruction Cache Tag RAM DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 1600$_H$ | ITECCCTL_PE2*1 | Instruction Cache Tag RAM ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP2 | — |
| FFC6 1604$_H$ | ITERRINT_PE2*1 | Instruction Cache Tag RAM Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 1608$_H$ | ITSERSTCLR_PE2*1 | Instruction Cache Tag RAM ECC SED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 160C$_H$ | ITDERSTCLR_PE2*1 | Instruction Cache Tag RAM ECC DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 1610$_H$ | ITOVFSTR_PE2*1 | Instruction Cache Tag RAM Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |

Table 28.57 List of Registers (3/3)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 $1620_H$ | ITSERSTR_PE2[*1] | Instruction Cache Tag RAM ECC SED Status Register (PE2) | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 $1650_H$ | ITDERSTR_PE2[*1] | Instruction Cache Tag RAM ECC DED Status Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 $1660_H$ | ITSEDADR_PE2[*1] | Instruction Cache Tag RAM SED Address Register | R | 0000 $0000_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 $16E0_H$ | ITDEDADR_PE2[*1] | Instruction Cache Tag RAM DED Address Register | R | 0000 $0000_H$ | 32 | APBGRD_ PFSS1.SP2 | — |

Note 1. P1H-CE and P1H-C devices only. P1M-C device doesn't have these registers.

### 28.2.6.3 Details of Registers

#### (1) IDECCCTL_PE1/PE2 — Instruction Cache Data RAM Data ECC Control Register

IDECCCTL_PE1/PE2 registers control the ECC error detection. Writing ECC control registers can only be executed with PROT[1:0] = $01_B$.

**Access:** IDECCCTL_PE1/PE2 can be read/written in 32/16-bit units.

**Address:** IDECCCTL_PE1: FFC6 $0400_H$
IDECCCTL_PE2: FFC6 $0600_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT[1:0] | | — | — | — | — | — | — | — | — | — | — | — | — | — | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 28.58 IDECCCTL_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | ECCDIS | ECC disable bit<br>Setting ECC error detection to enable/disable.<br>0: ECC error detection is enable<br>1: ECC error detection is disable |

### (2) IDERRINT_PE1/PE2 — Instruction Cache Data RAM Error Information Control Register

IDERRINT_PE1/PE2 registers controls whether error information is reported to ECM, when data ECC 2-bit error and data ECC 1-bit error are detected.

**Access:** IDERRINT_PE1/PE2 can be read/written in 32/16/8-bit units.

**Address:** IDERRINT_PE1: FFC6 0404$_H$
IDERRINT_PE2: FFC6 0604$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.59 IDERRINT_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | ECC 2-bit error notification enable bit<br>Control error notification of 2-bit error detection when ECC error detection is enabled<br>0: ECC 2-bit error notification disabled<br>1: ECC 2-bit error notification enabled |
| 0 | SEDIE | ECC 1-bit error notification enable bit<br>Control error notification of 1-bit error detection when ECC error detection is enabled<br>0: ECC 1-bit error notification disabled<br>1: ECC 1-bit error notification enabled |

### (3) IDSERSTCLR_PE1/PE2 — Instruction Cache Data RAM Data ECC SED Status Clear Register

IDSERSTCLR_PE1/PE2 registers are used to clear SEDF0/1 in IDSERSTR_PE1/PE2 and SERROVF0/1 in IDOVFSTR_PE1/PE2, and error address in IDSEDADR0/1_PE1/PE2. These are write only registers and read value is always “0”.

**Access:** IDSERSTCLR_PE1/PE2 can be written in 32/16/8-bit units.

**Address:** IDSERSTCLR_PE1: FFC6 $0408_H$
IDSERSTCLR_PE2: FFC6 $0608_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR1 | SSTCLR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W | W |

**Table 28.60 IDSERSTCLR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When writing, write the value after reset. |
| 1 | SSTCLR1 | 1-bit error flag clear (bank 1)<br>1: All 1-bit error flag clear<br>IDSERSTR_PE1/PE2.SEDF1,<br>IDOVFSTR_PE1/PE2.SERROVF1,<br>IDSEDADR1_PE1/PE2.SEADR |
| 0 | SSTCLR0 | 1-bit error flag clear (bank 0)<br>1: All 1-bit error flag clear<br>IDSERSTR_PE1/PE2.SEDF0,<br>IDOVFSTR_PE1/PE2.SERROVF0,<br>IDSEDADR0_PE1/PE2.SEADR |

### (4) IDDERSTCLR_PE1/PE2 — Instruction Cache Data RAM Data ECC DED Status Clear Register

IDDERSTCLR_PE1/PE2 register is used to clear DEDF0/1 in IDDERSTR_PE1/PE2 and DERROVF0/1 in IDOVFSTR_PE1/PE2, and error address in IDDEDADR0/1_PE1/PE2. This is write only register and read value is always “0”.

**Access:** IDDERSTCLR_PE1/PE2 can be written in 32/16/8-bit units.

**Address:** IDDERSTCLR_PE1: FFC6 040C$_H$
IDDERSTCLR_PE2: FFC6 060C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DSTCLR1 | DSTCLR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W | W |

**Table 28.61 IDDERSTCLR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When writing, write the value after reset. |
| 1 | DSTCLR1 | 2-bit error flag clear (bank 1)<br>1: All 2-bit error flag clear<br>IDDERSTR_PE1/PE2.DEDF1,<br>IDOVFSTR_PE1/PE2.DERROVF1<br>IDDEDADR1_PE1/PE2.DEADR |
| 0 | DSTCLR0 | 2-bit error flag clear (bank 0)<br>1: All 2-bit error flag clear<br>IDDERSTR_PE1/PE2.DEDF0,<br>IDOVFSTR_PE1/PE2.DERROVF0,<br>IDDEDADR0_PE1/PE2.DEADR |

### (5) IDOVFSTR_PE1/PE2 — Instruction Cache Data RAM Error Count Overflow Status Register

IDOVFSTR_PE1/PE2 registers monitor if error overflow occurs. Overflow occurs when different overflow*1 is detected in the case that error status is full. If the error status is full and the same error (same error cause and same error address) has occurred, then this flag is not set. SERROVF0/1, DERROVF0/1 flag is cleared by system reset or SSTCLR0/1, DSTCLR0/1 in IDSERSTCLR_PE1/PE2, IDDERSTCLR_PE1/PE2 register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different.
- Detected error status flag is different.

**Access:** IDOVFSTR_PE1/PE2 can be read in 32/16/8-bit units.

**Address:** IDOVFSTR_PE1: FFC6 0410$_H$
IDOVFSTR_PE2: FFC6 0610$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | DERROVF1 | DERROVF0 | — | — | SERROVF1 | SERROVF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.62 IDOVFSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. |
| 5 | DERROVF1 | 2-bit error overflow flag (bank1)<br>This error flag is set if the followings occur.<br>• DEDF1 in IDDERSTR_PE1/PE2 is set.<br>• Error with neither the same error cause nor the same address occur. |
| 4 | DERROVF0 | 2-bit error overflow flag (bank0)<br>This error flag is set if the followings occur.<br>• DEDF0 in IDDERSTR_PE1/PE2 is set.<br>• Error with neither the same error cause nor the same address occur. |
| 3 to 2 | Reserved | When read, the value after reset is read. |
| 1 | SERROVF1 | 1-bit error overflow flag (bank1)<br>This error flag is set if the followings occur.<br>• SEDF1 in IDSERSTR_PE1/PE2 is set.<br>• Error with neither the same error cause nor the same address occur. |
| 0 | SERROVF0 | 1-bit error overflow flag (bank0)<br>This error flag is set if the followings occur.<br>• SEDF0 in IDSERSTR_PE1/PE2 is set.<br>• Error with neither the same error cause nor the same address occur. |

### (6) IDSERSTR_PE1/PE2 — Instruction Cache Data RAM Data ECC SED Status Register

IDSERSTR_PE1/PE2 is the error monitor register. All error flag is “0” for each bank and a new error occurs, the error status flag is set. IDSERSTR_PE1/PE2 register is cleared by system reset or SSTCLR1/0 in IDSERSTCLR_PE1/PE2 register.

**Access:** IDSERSTR_PE1/PE2 can be read in 32/16/8-bit units.

**Address:** IDSERSTR_PE1: FFC6 $0420_H$
IDSERSTR_PE2: FFC6 $0620_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SEDF1 | — | — | — | — | — | — | — | SEDF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.63 IDSERSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is read. |
| 8 | SEDF1 | ECC 1-bit error monitor flag (bank 1)<br>Condition for “0”: system reset or write “1” to SSTCLR1 in IDSERSTCLR_PE1/PE2<br>Condition for “1”: SEDF1 is “0” and ECC 1-bit error is detected. |
| 7 to 1 | | When read, the value after reset is read. |
| 0 | SEDF0 | ECC 1-bit error monitor flag (bank 0)<br>Condition for “0”: system reset or write “1” to SSTCLR0 in IDSERSTCLR_PE1/PE2<br>Condition for “1”: SEDF0 is “0” and ECC 1-bit error is detected. |

### (7) IDDERSTR_PE1/PE2 — Instruction Cache Data RAM Data ECC DED Status Register

IDDERSTR_PE1/PE2 is the error monitor register. If an error flag is “0” for a specific bank and a new ECC 2-bit error occurs by reading from this bank, then the error status flag is set. IDDERSTR_PE1/PE2 register is cleared by system reset or DSTCLR1/0 in IDDERSTCLR_PE1/PE2 register.

**Access:** IDDERSTR_PE1/PE2 can be read in 32/16/8-bit units.

**Address:** IDDERSTR_PE1: FFC6 $0450_H$
IDDERSTR_PE2: FFC6 $0650_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | DEDF1 | — | — | — | — | — | — | — | DEDF0 | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.64 IDDERSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 10 | Reserved | When read, the value after reset is read. |
| 9 | DEDF1 | ECC 2-bit error monitor flag (bank 1)<br>Condition to “0”: system reset or write “1” to DSTCLR1 in IDDERSTCLR_PE1/PE2<br>Condition to “1”: DEDF1 is “0” and ECC 2-bit error is detected. |
| 8 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF0 | ECC 2-bit error monitor flag (bank 0)<br>Condition to “0”: system reset or write “1” to DSTCLR0 in IDDERSTCLR_PE1/PE2<br>Condition to “1”: DEDF0 is “0” and ECC 2-bit error is detected. |
| 0 | Reserved | When read, the value after reset is read. |

### (8) IDSEDADR0/1_PE1/PE2 — Instruction Cache Data RAM (Bank0/1) Data ECC SED Address Register

IDSEDADR0/1_PE1/PE2 registers are used to hold the address when an SED error is detected. Error address capture trigger is same as corresponding SEDF0/1 set in IDSERSTR_PE1/PE2. IDSEDADR0/1_PE1/PE2 are cleared by system reset or SSTCLR1/0 in IDSERSTCLR_PE1/PE2 register.

**Access:** IDSEDADR0/1_PE1/PE2 can be read in 32-bit units.

**Address:** IDSEDADR0_PE1: FFC6 0460$_H$
IDSEDADR1_PE1: FFC6 0464$_H$
IDSEDADR0_PE2: FFC6 0660$_H$
IDSEDADR1_PE2: FFC6 0664$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SEADR[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.65 IDSEDADR0/1_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is read. |
| 8 to 0 | SEADR[8:0] | 1-bit error detection address bank n (n = 0 or 1) |

### (9) IDDEDADR0/1_PE1/PE2 — Instruction Cache Data RAM (Bank0/1) Data ECC DED Address Register

IDDEDADR0/1_PE1/PE2 register is used to hold the address when a DED error is detected. Error address capture trigger is same as corresponding DEDF0/1 set in IDDERSTR_PE1/PE2. IDDEDADR0/1_PE1/PE2 is cleared by system reset or DSTCLR0/1 in IDDERSTCLR_PE1/PE2 register.

**Access:** IDDEDADR0/1_PE1/PE2 can be read in 32-bit units.

**Address:** IDDEDADR0_PE1: FFC6 04E0$_H$
IDDEDADR1_PE1: FFC6 04E4$_H$
IDDEDADR0_PE2: FFC6 06E0$_H$
IDDEDADR1_PE2: FFC6 06E4$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DEADR[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.66 IDDEDADR0/1_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is read. |
| 8 to 0 | DEADR[8:0] | 2-bit error detection address bank n (n = 0 or 1) |

### (10) ITECCCTL_PE1/PE2 — Instruction Cache Tag RAM ECC Control Register

ITECCCTL_PE1/PE2 register control the ECC error detection. Writing ECC control registers can only be executed with PROT[1:0] = $01_B$.

**Access:** ITECCCTL_PE1/PE2 can be read/written in 32/16-bit units.

**Address:** ITECCCTL_PE1: FFC6 $1400_H$
ITECCCTL_PE2: FFC6 $1600_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT[1:0] | | — | — | — | — | — | — | — | — | — | — | — | — | — | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 28.67 ITECCCTL_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | ECCDIS | ECC disable bit<br>Setting ECC error detection to enable/disable.<br>0: ECC error detection is enable<br>1: ECC error detection is disable |

### (11) ITERRINT_PE1/PE2 — Instruction Cache Tag RAM Error Information Control Register

ITERRINT_PE1/PE2 register controls whether error information is reported to ECM, when data ECC 2-bit error and data ECC 1-bit error are detected.

**Access:** ITERRINT_PE1/PE2 can be read/written in 32/16/8-bit units.

**Address:** ITERRINT_PE1: FFC6 1404$_H$
ITERRINT_PE2: FFC6 1604$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.68 ITERRINT_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when ECC error detection is enabled<br>0: ECC 2-bit error report disabled<br>1: ECC 2-bit error report enabled |
| 0 | SEDIE | ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when ECC error detection is enabled<br>0: ECC 1-bit error report disabled<br>1: ECC 1-bit error report enabled |

### (12) ITSERSTCLR_PE1/PE2 — Instruction Cache Tag RAM ECC SED Status Clear Register

ITSERSTCLR_PE1/PE2 registers are used to clear SEDF in ITSERSTR_PE1/PE2 and SERROVF in ITOVFSTR_PE1/PE2, and error address in ITSEDADR_PE1/PE2. These are write only registers and read value is always “0”.

**Access:** ITSERSTCLR_PE1/PE2 can be written in 32/16/8-bit units.

**Address:** ITSERSTCLR_PE1: FFC6 $1408_H$
ITSERSTCLR_PE2: FFC6 $1608_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.69 ITSERSTCLR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SSTCLR | 1-bit error flag clear<br>1: All 1-bit error flag clear<br>ITSERSTR_PE1/PE2.SEDF,<br>ITOVFSTR_PE1/PE2.SERROVF,<br>ITSEDADR_PE1/PE2.SEADR |

### (13) ITDERSTCLR_PE1/PE2 — Instruction Cache Tag RAM ECC DED Status Clear Register

ITDERSTCLR_PE1/PE2 register is used to clear DEDF in ITDERSTR_PE1/PE2, and DERROVF in ITOVFSTR_PE1/PE2, error address in ITDEDADR_PE1/PE2. This is write only register and read value is always “0”.

**Access:** ITDERSTCLR_PE1/PE2 can be written in 32/16/8-bit units.

**Address:** ITDERSTCLR_PE1: FFC6 140C$_H$
ITDERSTCLR_PE2: FFC6 160C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.70 ITDERSTCLR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | DSTCLR | 2-bit error flag clear (bank 0)<br>1: All 2-bit error flag clear<br>ITDERSTR_PE1/PE2.DEDF,<br>ITOVFSTR_PE1/PE2.DERROVF,<br>ITDEDADR_PE1/PE2.DEADR |

### (14) ITOVFSTR_PE1/PE2 — Instruction Cache Tag RAM Error Count Overflow Status Register

ITOVFSTR_PE1/PE2 register monitor if error overflow occurs. Overflow occurs when different overflow*1 is detected in the case that error status is full. If the error status is full and the same error (same error cause and same error address) has occurred, then this flag is not set. SERROVF , DERROVF flag is cleared by system reset or SSTCLR , DSTCLR in ITSERSTCLR_PE1/PE2, ITDERSTCLR_PE1/PE2 register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** ITOVFSTR_PE1/PE2 can be read in 32/16/8-bit units.

**Address:** ITOVFSTR_PE1: FFC6 1410$_H$
ITOVFSTR_PE2: FFC6 1610$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | DERROVF | — | — | — | SERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.71 ITOVFSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is read. |
| 4 | DERROVF | 2-bit error overflow flag<br>This error flag is set if the followings occur.<br>• DEDF in ITDERSTR_PE1/PE2 is set.<br>• Error with neither the same error cause nor the same address occur. |
| 3 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SERROVF | 1-bit error overflow flag<br>This error flag is set if the followings occur.<br>• SEDF in ITSERSTR_PE1/PE2 is set.<br>• Error with neither the same error cause nor the same address occur. |

### (15) ITSERSTR_PE1/PE2 — Instruction Cache Tag RAM ECC SED Status Register

ITSERSTR_PE1/PE2 is the error monitor register for SED. Error flag is “0” and a new error occurs, error status flag is set. ITSERSTR_PE1/PE2 register is cleared by system reset or SSTCLR in ITSERSTCLR_PE1/PE2 register.

**Access:** ITSERSTR_PE1/PE2 can be read only in 32-bit units.

**Address:** ITSERSTR_PE1: FFC6 1420$_H$
ITSERSTR_PE2: FFC6 1620$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.72 ITSERSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | SEDF | ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in ITSERSTCLR_PE1/PE2<br>Condition to “1”: SEDF is “0” and ECC 1-bit error is detected. |

### (16) ITDERSTR_PE1/PE2 — Instruction Cache Tag RAM ECC DED Status Register

ITDERSTR_PE1/PE2 is the DED error monitor register. Error flag is “0” and a new error occurs, error status flag is set. ITDERSTR_PE1/PE2 register is cleared by system reset or DSTCLR in ITDERSTCLR_PE1/PE2 register.

**Access:** ITDERSTR_PE1/PE2 can be read in 32/16/8-bit units.

**Address:** ITDERSTR_PE1: FFC6 $1450_H$
ITDERSTR_PE2: FFC6 $1650_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDF | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.73 ITDERSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF | ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to DSTCLR in ITDERSTCLR_PE1/PE2<br>Condition to “1”: DEDF is “0” and ecc 2-bit error is detected. |
| 0 | Reserved | When read, the value after reset is read. |

### (17) ITSEDADR_PE1/PE2 — Instruction Cache Tag RAM SED Address Register

ITSEDADR_PE1/PE2 register is used to hold the address when an SED error is detected. Error address capture trigger is same as corresponding SEDF set in ITSERSTR_PE1/PE2. ITSEDADR_PE1/PE2 is cleared by system reset or SSTCLR in ITSERSTCLR_PE1/PE2 register.

**Access:** ITSEDADR_PE1/PE2 can be read in 32-bit units.

**Address:** ITSEDADR_PE1: FFC6 $1460_H$
ITSEDADR_PE2: FFC6 $1660_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SEADR[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.74 ITSEDADR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is read. |
| 8 to 0 | SEADR[8:0] | 1-bit error detection address |

### (18) ITDEDADR_PE1/PE2 — Instruction Cache Tag RAM DED Address Register

ITDEDADR_PE1/PE2 register is used to hold the address when a DED error is detected. Error address capture trigger is same as corresponding DEDF set in ITDERSTR_PE1/PE2. ITDEDADR_PE1/PE2 is cleared by system reset or DSTCLR in ITDERSTCLR_PE1/PE2 register.

**Access:** ITDEDADR_PE1/PE2 can be read in 32-bit units.

**Address:** ITDEDADR_PE1: FFC6 $14E0_H$
ITDEDADR_PE2: FFC6 $16E0_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DEADR[8:0] | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.75 ITDEDADR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 9 | Reserved | When read, the value after reset is read. |
| 8 to 0 | DEADR[8:0] | 2-bit error detection address |

## 28.2.6.4 Test Function

A cache instruction is used to write the desired values as RAM data and to the ECC bits, and read data in the RAM and the ECC bits directly.

Since instructions as described above go through the same encoding or decoding path as a normal cache fill or instruction fetch, only such instructions can be used in inserting and confirming errors.

For details, see RH850/P1x-C Series User's Manual: Software.

### 28.2.7 DTS RAM ECC

See **Section 7, DMA**.

### 28.2.8 ECC for Peripheral RAM (32 Bits)

#### 28.2.8.1 Overview

This is an ECC module for the RAM of the following peripheral modules.

FlexRay, CSIHn (n = 0 to 3), Ethernet, MCAN, GTM

**(1) CSIHn (n = 0 to 3), MCAN, GTM**

**Error Detection and Correction**

Seven-bit ECC data is appended to the 32-bit RAM data.

This ECC circuit provides ECC 2-bit error detection and ECC 1-bit error detection and correction.

**Enabling or Disabling ECC Error Detection and Correction**

- ECC error detection can be either enabled or disabled.
- 1-bit ECC error correction can be either enabled or disabled.
- If all the bits of RAM output data are stuck to 0 or 1, it is detected as an ECC 2-bit error.

**ECM error notifications**

- An error notification is issued upon detection of an ECC 2-bit error (issuance can be either enabled or disabled).
- An error notification is issued upon detection of an ECC 1-bit error (issuance can be either enabled or disabled).

Once an error notification is issued, another error notification is not issued until the corresponding error status is cleared even if another ECC error is detected.

**Error Status**

- Detection of ECC 2- and 1-bit errors can be monitored.
- Special registers are provided to clear error status.

**Address Capture**

- Only one address at which an ECC error has occurred can be captured.
- A signal is generated upon detection of ECC 2-bit or 1-bit error, and the signal is used as a trigger to capture the error-causing address (when the first (1-bit or 2-bit) error is detected after the flag is cleared).

**Testing Function (Error Injection)**

- By setting the test mode, register values can be used as the data to be output to the RAM. When a peripheral module writes to the RAM, the ECEDB[31:0] register value can be written to the RAM data section, and the ECERDB[6:0] register value can be written to the ECC bit section.

- By setting the test mode, the ECC bit section can be latched when RAM data is read, and the value can be confirmed.
- By setting the test mode, the ECC bit (encoding circuit) and syndrome code (decoding circuit), which are generated from the input data, can be confirmed.

### (2) Ethernet

#### Error Detection and Correction

Seven-bit ECC data is appended to the 32-bit RAM data.

This ECC circuit provides ECC 2-bit error detection and ECC 1-bit error detection and correction.

#### Enabling or Disabling ECC Error Detection and Correction

To enable or disable an ECC error detection is not selectable; this function is always enabled.

To enable or disable an ECC 1-bit error correction is not selectable; this function is always enabled.

#### ECM error notifications

- An error notification is generated upon detection of an ECC 2-bit error (generation can't be disabled).
- An error notification is generated upon detection of an ECC 1-bit error (generation can't be disabled).

#### Error Status

None

#### Address Capture

None

#### Testing Function (Error Injection)

None

### (3) FlexRay

#### Error Detection and Correction

Seven-bit ECC data is appended to the 32-bit RAM data.
This ECC circuit provides ECC 2-bit error detection and ECC 1-bit error detection and correction.

#### Enabling or Disabling ECC Error Detection and Correction

- ECC error detection can be either enabled or disabled.
- 1-bit ECC error correction can be either enabled or disabled.
- If all the bits of RAM output data are stuck to 0 or 1, it is detected as an ECC 2-bit error.

### ECM error notifications

- An error notification is issued upon detection of an ECC 2-bit error (issuance can be either enabled or disabled).
- An error notification is issued upon detection of an ECC 1-bit error (issuance can be either enabled or disabled).
  Once an error notification is issued, another error notification is not issued until the corresponding error status is cleared even if another ECC error is detected.

### Error Status

- Detection of ECC 2- and 1-bit errors can be monitored.
- Special registers are provided to clear error status.

### Address Capture

- Only one address at which an ECC error has occurred can be captured.
- A signal is generated upon detection of ECC 2-bit or 1-bit error, and the signal is used as a trigger to capture the error-causing address (when the first (1-bit or 2-bit) error is detected after the flag is cleared).

### Testing Function (Error Injection)

None

### 28.2.8.2 List of Registers

#### (1) List of ECC Modules

The RAMs of the multiple peripheral functions are provided with the ECC modules. The following table shows the peripheral functions provided with the ECC modules, the corresponding ECC module names, and base addresses of the ECC modules.

**Table 28.76 ECC Module List**

| Peripheral Functions | | Symbol | ECC Module Names and Register Base Addresses | | PBG | Other Protection |
|---|---|---|---|---|---|---|
| | | | Module Name | Base Address <base_addr> | | |
| FlexRay | Message RAM (MRAM) | ECCFLX0 | ECCFLX0 | FFC7 2000$_H$ | PBG3#0.PG3-Startup | — |
| | Temporary buffer (TBF A) | ECCFLX0T0 | ECCFLX0T0 | FFC7 2100$_H$ | PBG3#0.PG3-Startup | — |
| | Temporary buffer (TBF B) | ECCFLX0T1 | ECCFLX0T1 | FFC7 2200$_H$ | PBG3#0.PG3-Startup | — |
| | Message RAM (MRAM) | ECCFLX1 | ECCFLX1 | FFC7 2400$_H$ | PBG3#0.PG3-Startup | — |
| | Temporary buffer (TBF A) | ECCFLX1T0 | ECCFLX1T0 | FFC7 2500$_H$ | PBG3#0.PG3-Startup | — |
| | Temporary buffer (TBF B) | ECCFLX1T1 | ECCFLX1T1 | FFC7 2600$_H$ | PBG3#0.PG3-Startup | — |
| CSIH | CSIH0 | ECCCSIH0 | ECCCSIH0 | FFC7 0000$_H$ | PBG3#0.PG3-Startup | — |
| | CSIH1 | ECCCSIH1 | ECCCSIH1 | FFC7 8000$_H$ | PBG1#0.PG1-Startup | — |
| | CSIH2 | ECCCSIH2 | ECCCSIH2 | FFC7 0100$_H$ | PBG3#0.PG3-Startup | — |
| | CSIH3 | ECCCSIH3 | ECCCSIH3 | FFC7 8100$_H$ | PBG1#0.PG1-Startup | — |
| MCAN | ECCTTCAN0 | ECCTTCAN0 | ECCTTCAN0 | FFC7 1000$_H$ | PBG3#0.PG3-Startup | — |
| | ECCMCAN0 | ECCMCAN0 | ECCMCAN0 | FFC7 9000$_H$ | PBG1#0.PG1-Startup | — |
| | ECCMCAN1 | ECCMCAN1 | ECCMCAN1 | FFC7 1100$_H$ | PBG3#0.PG3-Startup | — |
| | ECCMCAN2 | ECCMCAN2 | ECCMCAN2 | FFC7 9100$_H$ | PBG1#0.PG1-Startup | — |
| GTM | ECCGTM0 | ECCGTM0 | ECCGTM0 | FFE8 0000$_H$ | PBG2.PG2-Startup | — |
| | ECCGTM1 | ECCGTM1 | ECCGTM1 | FFE8 0100$_H$ | PBG2.PG2-Startup | — |
| | ECCGTM2 | ECCGTM2 | ECCGTM2 | FFE8 0200$_H$ | PBG2.PG2-Startup | — |
| | ECCGTM3 | ECCGTM3 | ECCGTM3 | FFE8 0300$_H$ | PBG2.PG2-Startup | — |

### (2) List of Registers

Each ECC module has the registers shown in the following table.

**Table 28.77 List of Registers**

| Register Name | Additional Abbreviation*2 | R/W | Value after reset | Address | Access Size |
|---|---|---|---|---|---|
| ECC control register*1 | CTL | R/W | $001X_H$ | <base_addr> + $00_H$ | 32 |
| ECC test mode control register*3 | TMC | R/W | $0000_H$ | <base_addr> + $04_H$ | 32 |
| ECC bit data control test register*3 | TRC | R/W | $0000\ 0000_H$ | <base_addr> + $08_H$ | 32 |
| ECC encoder and decoder data test register*3 | TED | R/W | $0000\ 0000_H$ | <base_addr> + $0C_H$ | 32 |
| ECC error address register | EAD0 | R | $0000\ 0000_H$ | <base_addr> + $10_H$ | 32 |

Note 1. The reset value of the LSB in the ECC control register is undefined.

Note 2. “Additional Abbreviation” is added to the symbol in the list of ECC modules that correspond to peripheral functions. For example, ECCCSIH2TMC represents the ECC test mode control register of CSIH2.

Note 3. Except for FlexRay

### (3) Register Map

**Table 28.78 Register Map**

| Abbreviation | 31 24 | 23 16 | 15 8 | 7 0 | Address |
|---|---|---|---|---|---|
| CTL | — ($00_H$) | CTL[23:16] | CTL[15:8] | CTL[7:0] | $nn00_H$ |
| TMC | — ($00_H$) | — ($00_H$) | TMC[15:8] | TMC[7:0] | $nn04_H$ |
| TRC | SYND[7:0] | HORD[7:0] | ECRD[7:0] | ERDB[7:0] | $nn08_H$ |
| TED | ECEDB[31:24] | ECEDB[23:16] | ECEDB[15:8] | ECEDB[7:0] | $nn0C_H$ |
| EAD0 | ECEAD[31:24] | ECEAD[23:16] | ECEAD[15:8] | ECEAD[7:0] | $nn10_H$ |

### 28.2.8.3 Details of Registers

#### (1) CTL — ECC control register

The CTL register controls the mode of the ECC for target peripheral modules.

Bits 7, 5, 4 and 3 should be set (written) while the target peripheral module operation is stopped.

In addition, when writing to bit 7, EMCA1 and EMCA0 need to be $01_B$.

**Access:** This register can be read/written in 32-bit units.

**Address:** See **Table 28.76, ECC Module List** and **Table 28.78, Register Map**.

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ECDEDF0 | ECSEDF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EMCA1 | EMCA0 | — | — | ECOVF | ECER2C | ECER1C | — | ECTHM | — | EC1ECP | EC2EDIC | EC1EDIC | ECER2F | ECER1F | ECEMF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Un-defined |
| R/W | R/W*1 | R/W*1 | R | R | R | R/W*1 | R/W*1 | R | R/W | R | R/W | R/W | R/W | R | R | R |

Note 1. This bit is always read as 0.

**Table 28.79 CTL register contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 17 | ECDEDF0 | |
| 16 | ECSEDF0 | |

| ECDEDF0 | ECSEDF0 | Operation explanation |
|---|---|---|
| 0 | 0 | There is no error address in EAD0 after reset or clearing ECER2F and ECER1F bits.<br>Clearing at<br>(1) reset<br>(2) writing ECER2C = 1 or ECER1C = 1<br>(3) selecting through mode enable (ECTHM = 1) |
| 1 | 0 | Address captured in EAD0 shows that ECC 2-bit error has occurred and the related address is captured. |
| 0 | 1 | Address captured in EAD0 shows that ECC 1-bit error has occurred and the related address is captured. |
| 1 | 1 | Setting prohibited |

| | | |
|---|---|---|
| 15 | EMCA1 | Access control bits 1 and 0 to ECC mode selection bit<br>These bits specify whether updating the ECTHM bit (bit 7) is disabled or enabled. The value written to these bits is not retained. When these bits are read, the read value is always 0. When these bits are $01_B$, writing to bit 7 is enabled. |
| 14 | EMCA0 | |
| 13, 12 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

**Table 28.79 CTL register contents (2/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 11 | ECOVFF | By detecting an error while the error status is set and the new error has another address than the already latched (not cleared or reset is not issued), this bit is set and error notification is generated.<br>**ECOVFF** / **Operation explanation**<br>0: Overflow is not occurred after reset or clearing ECER2F and ECER1F. Clearing at<br>(1) reset<br>(2) writing ECER2C = 1 or ECER1C = 1<br>(3) selecting through mode enable (ECTHM = 1)<br>1: Error address register overflowed. |
| 10 | ECER2C | ECC 2-bit error detection flag clear bit<br>This bit is used to clear bit 2, the status flag (ECER2F).<br>This bit is always read as 0.<br>Write 1 to this bit while the ECER2F bit is set to clear the ECER2F bit.<br>When a conflict between this bit writing and ECER2F bit setting occurs, writing to this bit has a priority. |
| 9 | ECER1C | ECC 1-bit error detection correction accumulation flag clear bit<br>This bit is used to clear bit 1, the status flag (ECER1F). This bit is always read as 0.<br>Write 1 to this bit while the ECER1F bit is set to clear the ECER1F bit.<br>When a conflict between this bit writing and ECER1F bit setting occurs, writing to this bit has a priority. |
| 8 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 7 | ECTHM | ECC function through mode selection bit<br>Set this bit to select whether to pass through the function of the ECC decoder. When writing to this bit, (0, 1) should be written to (EMCA1, EMCA0) at the same time. Set this bit to 1 to disable the ECC function.<br>0: Passing through mode is disabled (normal operation mode).<br>1: Passing through mode is enabled.<br>The encoder is not affected.<br>The decoder stops error detection and bit correction. |
| 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 | EC1ECP | ECC 1-bit error correction enable bit<br>This bit specifies whether to enable or disable 1-bit error correction when the ECC error detection/correction is enabled.<br>0: When 1-bit error is detected, the error will be corrected.<br>1: When 1-bit error is detected, the error will not be corrected. |
| 4 | EC2EDIC | ECC 2-bit error detection error notification control bit<br>This bit controls whether to generate an error notification when 2-bit error is detected.<br>0: When 2-bit error is detected, an error notification will not be generated.<br>1: When 2-bit error is detected, an error notification will be generated. |
| 3 | EC1EDIC | ECC 1-bit error detection error notifications control bit<br>This bit controls whether to generate an error notification when 1-bit error is detected.<br>0: When 1-bit error is detected, an error notification will not be generated.<br>1: When 1-bit error is detected, an error notification will be generated. |
| 2 | ECER2F | ECC 2-bit error detection flag bit<br>This flag indicates whether 2-bit error is detected during read access to the RAM when error determination is enabled. When 2-bit error notification is enabled and this flag is set, a 2-bit error notification signal is output. Write 1 to the ECER2C bit (bit 10) to clear the flag. If 2-bit error is detected again while this bit is set, an error notification request will not be generated.<br>0: 2-bit error has not occurred since this bit was cleared.<br>1: 2-bit error has occurred. |

Table 28.79 CTL register contents (3/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | ECER1F | 1-bit error detection/correction flag bit<br>This flag indicates whether 1-bit error is detected during read access to the RAM when error determination is enabled. Write 1 to the ECER1C bit (bit 9) to clear the flag.<br>0: 1-bit error has not occurred since this bit was cleared.<br>1: 1-bit error has occurred. |
| 0 | ECEMF | ECC error message flag<br>This flag indicates whether an error exists in the current read data bus. This bit is updated whenever the RAM outputs data. Because the value after reset of the RAM data is undefined, If the RAM is read before initialization, this bit may be set.<br>0: The current RAM output data does not have bit errors.<br>1: The current RAM output data have bit errors. |

**CAUTION**

Bits 2 and 1 should be cleared when the ECC error message flag (ECEMF) is not set.

We recommend initializing the RAM before clearing bits 2 and 1.

### (2) TMC — ECC test mode control register

The TMC register is used to switch to the test mode, and this register is for test mode.

When writing to bit 7, ETMA1 and ETMA0 need to be $10_B$.

**Access:** This register can be read/written in 32-bit units.

**Address:** See **Table 28.76, ECC Module List** and **Table 28.78, Register Map**.

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETMA1 | ETMA0 | — | — | — | — | — | — | ECTMCE | — | — | ECTRRS | ECREOS | ECENS | ECDCS | ECREIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W*1 | R/W*1 | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W |

Note 1. This bit is always read as 0.

**Table 28.80 TMC register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 | ETMA1 | Access control bits 1 and 0 to ECC test mode bit<br>These two bits specify whether updating the ECTMCE bit (bit 7) is disabled or enabled. The value written to these bits is not retained. When these bits are read, the read value is always 0.<br>When these bits are $10_B$, writing to bit 7 is enabled. |
| 14 | ETMA0 | |
| 13 to 8 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 7 | ECTMCE | ECC test mode enable bit<br>This bit specifies whether to enable access to the test registers and test control bits. When writing to this bit, (1, 0) should be written to (ETMA1, ETMA0) at the same time.<br>0: Access to the test mode registers and bits is disabled.<br>1: Access to the test mode registers and bits is enabled. |
| 6, 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 | ECTRRS | ECC RAM read test mode selection bit<br>This bit is used to generate a RAM read cycle. Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: A RAM read cycle is not generated even when TED is read.<br>1: A RAM read cycle is generated when TED is read.<br>In addition, the TED read value depends on the ECDCS bit (bit 1).<br>The ERDB read value depends on the ECREIS bit (bit 0). |
| 3 | ECREOS | ECC bit output data selection bit<br>This bit specifies which is output to the ECC bit output, the ECC encoder output data or the value of the ERDB register. Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously).<br>This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Encoding result is output to the ECC bit output.<br>1: TRC.ERDB[6:0] is output to the ECC bit output. |

**Table 28.80 TMC register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | ECENS | ECC encoder input selection bit<br>This bit specifies which is input to the encoder, the data from the peripheral macro or the value of the TED.ECEDB[31:0]. Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously).<br>This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Write data from the peripheral macro to the RAM is input to the ECC encoder as input data.<br>1: TED.ECEDB[31:0] is input to the ECC encoder as input data. |
| 1 | ECDCS | ECC decoder input selection bit<br>This bit specifies which is input to the decoder as the lower 32-bit data of the decoder input, the lower 32-bit data from RAM or the value of TED.ECEDB[31:0]. Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: The lower 32-bit data from the RAM is input to the data area (lower 32-bit data) to the decoder circuit.<br>1: TED.ECEDB[31:0] is input to the data area to the decoder circuit. |
| 0 | ECREIS | ECC bit input data selection bit<br>This bit specifies which is input as the upper 7-bit data of the decoder input, the upper 7-bit data (redundant bit area) from the RAM or the value of the TRC.ECERDB[6:0]. Writing to this bit is enabled only when ECTMCE = 1 (can be set simultaneously). This bit is cleared when ECTMCE = 0 (cleared synchronously).<br>0: Upper 7 bits of RAM output data are input to the ECC redundant bit area to the decoder circuit.<br>1: TRC.ECERDB[6:0] is input to the ECC redundant bit area to the decoder circuit. |

### (3) TED — ECC encoder and decoder data test register

TED is a test register for 32-bit data for ECC encoding/decoding.

In test mode, the value of this register can be used as input data for the encoder or decoder circuit.

**Access:** When TMC.ECTMCE = 1, this register can be read/written in 32-bit units.
When TMC.ECTMCE = 0, the value of this register is always $0000_H$.

**Address:** See **Table 28.76, ECC Module List** and **Table 28.78, Register Map**.

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEDB[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEDB[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.81 TED register contents**

| Bit position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECEDB[31:0] | When TMC.ECENS = 1, the value of this register is used as the input data to the encoder circuit and sent to the RAM.<br>When TMC.ECDCS = 1, the value of this register is used as the bit 31 to 0 of the input data to the decoder circuit.<br>In addition, when TMC.ECTRRS = 1, the read value from this register switches from the value of this register to the RAM output data. |

### (4) TRC — ECC redundant bit data control test register

This register is a 32-bit test register for the ECC redundant bit area and consists of four 8-bit registers, SYND, HORD, ECRD, and ERDB.

**Access:** When TMC.ECTMCE = 1, this register can be read/written in 32-bit units.
When TMC.ECTMCE = 0, the value of this register is always $0000_H$.

**Address:** See **Table 28.76, ECC Module List** and **Table 28.78, Register Map**.

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYND (See **(5)**) | | | | | | | | HORD (See **(6)**) | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECRD (See **(7)**) | | | | | | | | ERDB (See **(8)**) | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

### (5) SYND — ECC decoder syndrome data register

SYND is a read-only register to confirm the syndrome code generated by the decoding circuit in test mode (ECTMCE = 1).

Write access to SYND is ignored.

**Access:** When TMC.ECTMCE = 1, this register can be read only in 8-bit units.
When TMC.ECTMCE = 0, the value of this register is always $00_H$.

**Address:** See **Table 28.76, ECC Module List** and **Table 28.78, Register Map**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | SYND6 | SYND5 | SYND4 | SYND3 | SYND2 | SYND1 | SYND0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 28.82 SYND register contents**

| Bit position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is read. |
| 6 to 0 | SYND[6:0] | When reading this register bits, the syndrome code (synd[6:0]) generated in the decoder circuit based on the input data can be read.<br>The value of this register changes as the input data changes.<br>Note that this register is enabled only when ECTMCE = 1, and the value is always $00_H$ when ECTMCE = 0. |

### (6) HORD — ECC 7-Bit redundant data holding test register

HORD holds the 7-bit ECC redundant area (upper 7-bit RAM data), which cannot be confirmed by the peripheral module, when the peripheral module accesses the RAM for reading in test mode (ECTMCE = 1).

**Access:** When TMC.ECTMCE = 1, this register can be read only in 8-bit units.
When TMC.ECTMCE = 0, the value of this register is always $00_H$.

**Address:** See **Table 28.76, ECC Module List** and **Table 28.78, Register Map**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | HORD6 | HORD5 | HORD4 | HORD3 | HORD2 | HORD1 | HORD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 28.83 HORD register contents**

| Bit position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is read. |
| 6 to 0 | HORD[6:0] | When the peripheral module reads the RAM in test mode (ECTMCE = 1), this register fetches the upper 7 bits of the RAM output data.<br>In addition, if TMC.ECTRRS = 1, the value of the RAM output data is fetched to this register when the ECEDB[15:0] register is read.<br>Note that this register is enabled only when ECTMCE = 1. |

### (7) ECRD — ECC encoder test register

ECRD is a read-only register to read the 7-bit redundant section generated by the encoding circuit in test mode (ECTMCE = 1).

**Access:** When TMC.ECTMCE = 1, this register can be read only in 8-bit units.
When TMC.ECTMCE = 0, the value of this register is always $00_H$.

**Address:** See **Table 28.76, ECC Module List** and **Table 28.78, Register Map**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | ECRD6 | ECRD5 | ECRD4 | ECRD3 | ECRD2 | ECRD1 | ECRD0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 28.84 ECRD register contents**

| Bit position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is read. |
| 6 to 0 | ECRD[6:0] | ECRD is used to hold the ECC bits generated from the input data of the peripheral module. These bits can be checked to confirm correct encoding.<br>The read value is the encoding result (ecc[6:0]), and not the ECC redundant bit output value.<br>Note that this register is enabled only when ECTMCE = 1. |

### (8) ERDB — ECC bit input and output substitution buffer register

ERDB is a buffer register for the data that substitutes for the input and output data for the 7-bit ECC redundant data area in test mode (ECTMCE = 1).

ERDB can be read and written to in ECC test mode (ECTMCE = 1).

**Access:** When TMC.ECTMCE = 1, this register can be read/written in 8-bit units.
When TMC.ECTMCE = 0, the value of this register is always $00_H$.

**Address:** See **Table 28.76, ECC Module List** and **Table 28.78, Register Map**.

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | ERDB6 | ERDB5 | ERDB4 | ERDB3 | ERDB2 | ERDB1 | ERDB0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.85 ERDB register contents**

| Bit position | Bit Name | Function |
|---|---|---|
| 7 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 6 to 0 | ERDB[6:0] | When ECREOS = 1, the value of this resister is output to the ECC redundant bit output pins and sent to the RAM instead of the 7-bit long redundant bits generated by the encoder circuit.<br>When ECREIS = 1, the value of this register is used by the decoder circuit instead of the upper 7 bits of the data input to the decoder circuit.<br>In addition, when ECTRRS = 1, the read value from this register switches from the written value to this register to the RAM output data. |

### (9) EAD0 — ECC error address register

EAD0 is a read-only register to hold the address at which an ECC error has occurred.

**Access:** This register can be read in 32-bit units.

**Address:** See **Table 28.76, ECC Module List** and **Table 28.78, Register Map**.

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEAD[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEAD[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.86 EAD0 register contents**

| Bit position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECEAD[31:0] | EAD0 is a read-only register to hold the address at which an ECC error has occurred.<br>If an ECC error is detected while ECC error detection is enabled, the RAM address is latched using the detection signal as a trigger, and the address is stored in EAD0 as the address at which the ECC error has occurred.<br>The address is stored upon detection of the first ECC error while no error status is set. However, if a 1-bit error is followed by a 2-bit error, the address of the latter is stored.<br>Only one address can be held in EAD0. |

#### 28.2.8.4 Test Function

**(1) Writing RAM data**

Write data to the peripheral RAM. However, ECC corresponding to the write data is simultaneously written to the ECC bits. To write an arbitrary value to the ECC bits, use ECC test mode shown in (3).

**(2) Reading RAM data**

(a) Set the ECTHM bit in the ECC control register to 1 to disable ECC error detection/correction.

(b) Read the peripheral RAM data. The RAM data is directly read because error detection or correction is not performed during reading.

Exiting this test mode

(a) Set the ECTHM bit in the ECC control register to 0 to enable ECC error detection/correction.

**(3) Writing ECC bits**

(a) Set the ECTMCE bit in the ECC test mode control register to 1 to specify ECC test mode.

(b) Write a value to be written to the ECC bits to TRC.ERDB[6:0].

(c) Set the ECREOS bit in the ECC test mode control register to 1 to select TRC.ERDB[6:0] to be written to the ECC bits.

(d) When data is written to the peripheral RAM, the TRC.ERDB[6:0] value is written to the ECC bits.

Exiting this test mode

(a) Set the ECTMCE bit in the ECC test mode control register to 0 to specify normal mode.

**(4) Reading ECC bits**

(a) Set the ECTMCE bit in the ECC test mode control register to 1 to specify ECC test mode.

(b) When the peripheral RAM data is read, the value of the ECC bits is stored in TRC.HORD[6:0].

Exiting this test mode

(a) Set the ECTMCE bit in the ECC test mode control register to 0 to specify normal mode.

## 28.2.9 Safety Mechanism on Data Transfer Path

### 28.2.9.1 ECC protection on Bus

Below the target paths of ECC protection on bus are indicated. It is possible to detect an error of address and data line from an access master to an access slave. If ECC error is detected, error is signaled to ECM.

Table 28.87 ECC protection on Bus

| Access Source (Master) | Access Destination (Slave) |
|---|---|
| CPU1, CPU2, DMAC, DTS | GTM, ADCF, RAM of CPU1/2, Global RAM, all PBUS group except for path to AURORA |

Data protection targets have controller of ECC decoder and user can configure detection, signaling and so on. Error information is stored in status registers when error occurs.

In the case of an address error detection, even if detected error is 1 bit error, MCU suggests to have serious unrecoverable problem.

### 28.2.9.2 Safety Feature for GRAM Access

The GRAM IF (also known as GRAM Controller (GRAMC)), comprises the Write-Through-Buffer (WT-Buffer) for performance enhancements, the Read-Modify-Write (RmW) to allow various access bit-width (64 / 32 / 16 / 8 bit) and the arbiter for arbitration of several masters. See CPU section, GRAM Accesses for more explanation about user function.

Here, the safety mechanisms of the GRAMC are described:

1. WT-Buffer: The data in WT-Buffer is protected by data ECC, and other logics like Control, Address and Vld are redundant.
2. Address path: The address path of the data is protected by address ECC. The address ECC is generated inside the PE and evaluated at output of GRAMC.
3. RmW: For 8-bit and 16-bit write accesses, the data ECC has to be recomputed, hence the read modify write circuits are implemented redundant. The RmW circuits in before the arbiter do not comprise a state machine, therefore the RmW state checker is just implemented for the RmW circuit after the arbiter.
4. ReqBuff: The request buffers are protected by address ECC and data ECC.
5. Arbiter: The arbiter is implemented redundant.
6. RespBuff: The response buffers are protected by data ECC and the packet ID. If the ID of request packet doesn’t match with one of response packet, GRAM master cannot proceed to the next process. As a result, other safety mechanism such as watch dog timer for PE or E2E communication for HBUS peripherals will detect failure by expiration.

### 28.2.9.3 Fault Detection Mechanism of Arbitration

Bus arbiters in PFSS have arbiter check function which detects unintended arbiter status. Detected arbitration error is notified to ECM. Target arbiters are as follow:

**Table 28.88 Arbiter check**

| Target arbiter | Function |
|---|---|
| GRAMC | Global RAM arbiter is redundant. |
| FABTSS | Code Flash arbiter is redundant. |
| System interconnect | Arbitration check function |

### 28.2.9.4 List of Registers

**Table 28.89 List of Registers (1/8)**

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 2C00$_H$ | CFECCCTL_VCI2CFBB | FLI (Code-Flash) Address ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 2C04$_H$ | CFERRINT_VCI2CFBB | FLI (Code-Flash) Address Error Information Control Register | R/W | 0000 0030$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 2C08$_H$ | CFERSTCLR_VCI2CFBB | FLI (Code-Flash) Address ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 2C0C$_H$ | CFOVFSTR_VCI2CFBB | FLI (Code-Flash) Address Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 2C10$_H$ | CFERSTR_VCI2CFBB | FLI (Code-Flash) Address ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 2C50$_H$ | CFEADR0_VCI2CFBB | FLI (Code-Flash) Address ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 3400$_H$ | IFECCCTL_PE1 | IFU Data ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP1 | — |
| FFC6 3404$_H$ | IFERRINT_PE1 | IFU Data Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 3408$_H$ | IFERSTCLR_PE1 | IFU Data ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 340C$_H$ | IFOVFSTR_PE1 | IFU Data Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 3410$_H$ | IFERSTR_PE1 | IFU Data ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 3450$_H$ | IFEADR_PE1 | IFU Data ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 3600$_H$ | IFECCCTL_PE2[*1] | IFU Data ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP2 | — |
| FFC6 3604$_H$ | IFERRINT_PE2[*1] | IFU Data Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 3608$_H$ | IFERSTCLR_PE2[*1] | IFU Data ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |

Table 28.89 List of Registers (2/8)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 360C$_H$ | IFOVFSTR_PE2[*1] | IFU Data Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 3610$_H$ | IFERSTR_PE2[*1] | IFU Data ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 3650$_H$ | IFEADR_PE2[*1] | IFU Data ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 6000$_H$ | LSSECCCTL_PE1 | LSU Slave Data and Address ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP1 | — |
| FFC6 6004$_H$ | LSSERRINT_PE1 | LSU Slave Data and Address Error Information Control Register | R/W | 0000 0033$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 6008$_H$ | LSSERSTCLR_PE1 | LSU Slave Data and Address ECC SED/DED Status Clear Register | R/W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 600C$_H$ | LSSOVFSTR_PE1 | LSU Slave Data and Address Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 6010$_H$ | LSSERSTR_PE1 | LSU Slave Data and Address ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 6050$_H$ | LSSEADR_PE1 | LSU Slave Data and Address ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 6200$_H$ | LSSECCCTL_PE2[*1] | LSU Slave Data and Address ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP2 | — |
| FFC6 6204$_H$ | LSSERRINT_PE2[*1] | LSU Slave Data and Address Error Information Control Register | R/W | 0000 0033$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 6208$_H$ | LSSERSTCLR_PE2[*1] | LSU Slave Data and Address ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 620C$_H$ | LSSOVFSTR_PE2[*1] | LSU Slave Data and Address Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 6210$_H$ | LSSERSTR_PE2[*1] | LSU Slave Data and Address ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 6250$_H$ | LSSEADR_PE2[*1] | LSU Slave Data and Address ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 6400$_H$ | LSMECCCTL_PE1 | LSU Master Data ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP1 | — |
| FFC6 6404$_H$ | LSMERRINT_PE1 | LSU Master Data Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 6408$_H$ | LSMERSTCLR_PE1 | LSU Master Data ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 640C$_H$ | LSMOVFSTR_PE1 | LSU Master Data Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |

Table 28.89 List of Registers (3/8)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 6410$_H$ | LSMERSTR_PE1 | LSU Master Data ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP1 | — |
| FFC6 6450$_H$ | LSMEADR_PE1 | LSU Master Data ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP1 | — |
| FFC6 6600$_H$ | LSMECCCTL_PE2*1 | LSU Master Data ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP2 | — |
| FFC6 6604$_H$ | LSMERRINT_PE2*1 | LSU Master Data Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 6608$_H$ | LSMERSTCLR_PE2*1 | LSU Master Data ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 660C$_H$ | LSMOVFSTR_PE2*1 | LSU Master Data Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 6610$_H$ | LSMERSTR_PE2*1 | LSU Master Data ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP2 | — |
| FFC6 6650$_H$ | LSMEADR_PE2*1 | LSU Master Data ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP2 | — |
| FFC6 7000$_H$ | VPECCCTL_SG0*2 | System interconnect (PBus I/F) Data and Address ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7004$_H$ | VPERRINT_SG0 | System interconnect (PBus I/F) Data and Address Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7008$_H$ | VPERSTCLR_SG0 | System interconnect (PBus I/F) Data and Address ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 700C$_H$ | VPOVFSTR_SG0 | System interconnect (PBus I/F) Data Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7010$_H$ | VPERSTR_SG0 | System interconnect (PBus I/F) Data ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7050$_H$ | VPEADR_SG0 | System interconnect (PBus I/F) Data ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7400$_H$ | VPECCCTL_SG1*2 | System interconnect (PBus I/F) Data ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7404$_H$ | VPERRINT_SG1 | System interconnect (PBus I/F) Data Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |

Table 28.89 List of Registers (4/8)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 7408$_H$ | VPERSTCLR_SG1 | System interconnect (PBus I/F) Data ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 740C$_H$ | VPOVFSTR_SG1 | System interconnect (PBus I/F) Data Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7410$_H$ | VPERSTR_SG1 | System interconnect (PBus I/F) Data ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7450$_H$ | VPEADR_SG1 | System interconnect (PBus I/F) Data ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7800$_H$ | VPECCCTL_SG2*2 | System interconnect (PBus I/F) Data ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7804$_H$ | VPERRINT_SG2 | System interconnect (PBus I/F) Data Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7808$_H$ | VPERSTCLR_SG2 | System interconnect (PBus I/F) Data ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 780C$_H$ | VPOVFSTR_SG2 | System interconnect (PBus I/F) Data Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7810$_H$ | VPERSTR_SG2 | System interconnect (PBus I/F) Data ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7850$_H$ | VPEADR_SG2 | System interconnect (PBus I/F) Data ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7C00$_H$ | VPECCCTL_SG3*2 | System interconnect (PBus I/F) Data ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7C04$_H$ | VPERRINT_SG3 | System interconnect (PBus I/F) Data Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7C08$_H$ | VPERSTCLR_SG3 | System interconnect (PBus I/F) Data ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7C0C$_H$ | VPOVFSTR_SG3 | System interconnect (PBus I/F) Data Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 7C10$_H$ | VPERSTR_SG3 | System interconnect (PBus I/F) Data ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |

Table 28.89 List of Registers (5/8)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 7C50$_H$ | VPEADR_SG3 | System interconnect (PBus I/F) Data ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 8400$_H$ | VPECCCTL_SG5*2 | System interconnect (PBus I/F) Data ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 8404$_H$ | VPERRINT_SG5 | System interconnect (PBus I/F) Data Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 8408$_H$ | VPERSTCLR_SG5 | System interconnect (PBus I/F) Data ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 840C$_H$ | VPOVFSTR_SG5 | System interconnect (PBus I/F) Data Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 8410$_H$ | VPERSTR_SG5 | System interconnect (PBus I/F) Data ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 8450$_H$ | VPEADR_SG5 | System interconnect (PBus I/F) Data ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 8E00$_H$ | VPECCCTL_TERM_SG7 *2 | System interconnect (PBus I/F) Data and Address ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 8E04$_H$ | VPERRINT_TERM_SG7 | System interconnect (PBus I/F) Data and Address Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 8E08$_H$ | VPERSTCLR_TERM_SG 7 | System interconnect (PBus I/F) Data and Address ECC SED/DED Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 8E0C$_H$ | VPOVFSTR_TERM_SG7 | System interconnect (PBus I/F) Data Error Count Overflow Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 8E10$_H$ | VPERSTR_TERM_SG7 | System interconnect (PBus I/F) Data and Address ECC SED/DED Status Register | R | 0000 0000$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 8E50$_H$ | VPEADR_TERM_SG7 | System interconnect (PBus I/F) Data ECC SED/DED Address Register | R | 0000 0000$_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFC6 9000$_H$ | VPECCCTL_VCI2VPI*2 | System interconnect Address ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 9004$_H$ | VPERRINT_VCI2VPI | System interconnect Address ECC Error Information Control Register | R/W | 0000 0003$_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |

Table 28.89 List of Registers (6/8)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFC6 $9008_H$ | VPERSTCLR_VCI2VPI | System interconnect Address ECC SED/DED Status Clear Register | W | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $900C_H$ | VPOVFSTR_VCI2VPI | System interconnect Address Error Count Overflow Status Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $9010_H$ | VPERSTR_VCI2VPI | System interconnect Address ECC SED/DED Status Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $9050_H$ | VPERADR0_VCI2VPI | System interconnect Address ECC SED/DED Address Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $A000_H$ | VCECCCTL_PDMA | System Interconnect Data ECC Control Register | R/W | 0000 $0000_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $A004_H$ | VCERRINT_PDMA | System Interconnect Data Error Information Control Register | R/W | 0000 $0003_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $A008_H$ | VCERSTCLR_PDMA | System Interconnect Data ECC SED/DED Status Clear Register | W | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $A00C_H$ | VCOVFSTR_PDMA | System Interconnect Data Error Count Overflow Status Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $A010_H$ | VCERSTR_PDMA | System Interconnect Data ECC SED/DED Status Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $A050_H$ | VCEADR_PDMA | System Interconnect Data ECC SED/DED Address Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $C000_H$ | APECCCTL_PFSS | P-Bus Data and Address ECC Control Register | R/W | 0000 $0000_H$ | 32/16 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $C004_H$ | APERRINT_PFSS | P-Bus Data and Address Error Information Control Register | R/W | 0000 $0073_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $C008_H$ | APERSTCLR_PFSS | P-Bus Data and Address ECC SED/DED Status Clear Register | W | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $C00C_H$ | APOVFSTR_PFSS | P-Bus Data and Address Error Count Overflow Status Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $C010_H$ | APERSTR_PFSS | P-Bus Data and Address ECC SED/DED Status Register | R | 0000 $0000_H$ | 32/16/8 | APBGRD_ PFSS1.SP4 | — |
| FFC6 $C050_H$ | APEADR_PFSS | P-Bus Data and Address ECC SED/DED Address Register | R | 0000 $0000_H$ | 32 | APBGRD_ PFSS1.SP4 | — |
| FFCB $8000_H$ | APEC0ECCCTL | ECC Control Register | R/W | 0000 $0000_H$ | 32/16 | PBG1#0.P G1-Startup | — |
| FFCB $8004_H$ | APEC0ERRINT | P-Bus Data and Address Error Information Control Register | R/W | 0000 $0073_H$ | 32/16/8 | PBG1#0.P G1-Startup | — |
| FFCB $8008_H$ | APEC0STCLR | Error Status Clear Register | W | 0000 $0000_H$ | 32/16/8 | PBG1#0.P G1-Startup | — |

Table 28.89 List of Registers (7/8)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFCB 800C$_H$ | APEC0OVFSTR | Error Overflow Register | R | 0000 0000$_H$ | 32/16/8 | PBG1#0.PG1-Startup | — |
| FFCB 8010$_H$ | APEC01STERSTR | 1st Error Status Register | R | 0000 0000$_H$ | 32/16/8 | PBG1#0.PG1-Startup | — |
| FFCB 8050$_H$ | APEC01STEADR0 | 1st Error Address Register0 | R | 0000 0000$_H$ | 32/16/8 | PBG1#0.PG1-Startup | — |
| FFE8 8000$_H$ | APEC1ECCCTL | ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | PBG2.PG2-Startup | — |
| FFE8 8004$_H$ | APEC1ERRINT | P-Bus Data and Address Error Information Control Register | R/W | 0000 0073$_H$ | 32/16/8 | PBG2.PG2-Startup | — |
| FFE8 8008$_H$ | APEC1STCLR | Error Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | PBG2.PG2-Startup | — |
| FFE8 800C$_H$ | APEC1OVFSTR | Error Overflow Register | R | 0000 0000$_H$ | 32/16/8 | PBG2.PG2-Startup | — |
| FFE8 8010$_H$ | APEC11STERSTR | 1st Error Status Register | R | 0000 0000$_H$ | 32/16/8 | PBG2.PG2-Startup | — |
| FFE8 8050$_H$ | APEC11STEADR0 | 1st Error Address Register0 | R | 0000 0000$_H$ | 32/16/8 | PBG2.PG2-Startup | — |
| FFE8 8200$_H$ | APEC2ECCCTL | ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | PBG2.PG2-Startup | — |
| FFE8 8204$_H$ | APEC2ERRINT | P-Bus Data and Address Error Information Control Register | R/W | 0000 0073$_H$ | 32/16/8 | PBG2.PG2-Startup | — |
| FFE8 8208$_H$ | APEC2STCLR | Error Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | PBG2.PG2-Startup | — |
| FFE8 820C$_H$ | APEC2OVFSTR | Error Overflow Register | R | 0000 0000$_H$ | 32/16/8 | PBG2.PG2-Startup | — |
| FFE8 8210$_H$ | APEC21STERSTR | 1st Error Status Register | R | 0000 0000$_H$ | 32/16/8 | PBG2.PG2-Startup | — |
| FFE8 8250$_H$ | APEC21STEADR0 | 1st Error Address Register0 | R | 0000 0000$_H$ | 32/16/8 | PBG2.PG2-Startup | — |
| FFF9 8000$_H$ | APEC3ECCCTL | ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | PBG3#0.PG3-Startup | — |
| FFF9 8004$_H$ | APEC3ERRINT | P-Bus Data and Address Error Information Control Register | R/W | 0000 0073$_H$ | 32/16/8 | PBG3#0.PG3-Startup | — |
| FFF9 8008$_H$ | APEC3STCLR | Error Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | PBG3#0.PG3-Startup | — |
| FFF9 800C$_H$ | APEC3OVFSTR | Error Overflow Register | R | 0000 0000$_H$ | 32/16/8 | PBG3#0.PG3-Startup | — |
| FFF9 8010$_H$ | APEC31STERSTR | 1st Error Status Register | R | 0000 0000$_H$ | 32/16/8 | PBG3#0.PG3-Startup | — |
| FFF9 8050$_H$ | APEC31STEADR0 | 1st Error Address Register0 | R | 0000 0000$_H$ | 32/16/8 | PBG3#0.PG3-Startup | — |
| FFCD 8000$_H$ | APEC4ECCCTL | ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | PBG4#0.PG4-Startup | — |
| FFCD 8004$_H$ | APEC4ERRINT | P-Bus Data and Address Error Information Control Register | R/W | 0000 0073$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 8008$_H$ | APEC4STCLR | Error Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |

Table 28.89 List of Registers (8/8)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection |
|---|---|---|---|---|---|---|---|
| FFCD 800C$_H$ | APEC4OVFSTR | Error Overflow Register | R | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 8010$_H$ | APEC41STERSTR | 1st Error Status Register | R | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 8050$_H$ | APEC41STEADR0 | 1st Error Address Register0 | R | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 8200$_H$ | APEC5ECCCTL | ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | PBG4#0.PG4-Startup | — |
| FFCD 8204$_H$ | APEC5ERRINT | P-Bus Data and Address Error Information Control Register | R/W | 0000 0073$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 8208$_H$ | APEC5STCLR | Error Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 820C$_H$ | APEC5OVFSTR | Error Overflow Register | R | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 8210$_H$ | APEC51STERSTR | 1st Error Status Register | R | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 8250$_H$ | APEC51STEADR0 | 1st Error Address Register0 | R | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 8400$_H$ | APEC6ECCCTL | ECC Control Register | R/W | 0000 0000$_H$ | 32/16 | PBG4#0.PG4-Startup | — |
| FFCD 8404$_H$ | APEC6ERRINT | P-Bus Data and Address Error Information Control Register | R/W | 0000 0073$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 8408$_H$ | APEC6STCLR | Error Status Clear Register | W | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 840C$_H$ | APEC6OVFSTR | Error Overflow Register | R | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 8410$_H$ | APEC61STERSTR | 1st Error Status Register | R | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |
| FFCD 8450$_H$ | APEC61STEADR0 | 1st Error Address Register0 | R | 0000 0000$_H$ | 32/16/8 | PBG4#0.PG4-Startup | — |

Note 1. P1H-CE and P1H-C devices only. P1M-C device doesn't have these registers.

Note 2. This register doesn't have any functions.

CFECCCTL_* and CFSTSCTL_* indicate control registers on each access port: "_VCI2CFBA" represents access from the system interconnect to the Code Flash, "_PE1", and "_PE2" represents access from the CPU1 respectively CPU2 to the Code Flash.

### 28.2.9.5 Details of Registers

#### (1) CFECCCTL_VCI2CFBB — FLI (Code-Flash) Address ECC Control Register

CFECCCTL_VCI2CFBB registers control the address ECC error detection/correction and 1-bit error correction on bus width conversion between code flash interface and system interconnection. Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

**Access:** These registers can be read/written in 32/16-bit units.

**Address:** FFC6 $2C00_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | ASECDIS | AECCDIS | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R |

**Table 28.90 CFECCCTL_VCI2CFBB register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 | ASECDIS | Address ECC 1-bit error correction enable bit<br>When using Address ECC error detection/correction (AECCDIS = 0), setting 1-bit error<br>correction to enable/disable.<br>0: Correction is carried out when 1-bit is detected<br>1: Correction is not carried out when 1-bit is detected |
| 2 | AECCDIS | Address ECC disable bit<br>Setting Address ECC error detection/correction to enable/disable.<br>0: Address ECC error detection/correction is enable<br>1: Address ECC error detection/correction is disable |
| 1, 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (2) CFERRINT_VCI2CFBB — FLI (Code-Flash) Address Error Information Control Register

CFERRINT_VCI2CFBB register controls whether error information is reported to ECM, when address ECC 2-bit error and address ECC 1-bit error are detected.

**Access:** These registers can be read/written in 32/16/8-bit units.

**Address:** FFC6 2C04$_H$

**Value after reset:** 0000 0030$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ADEDIE | ASEDIE | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R | R |

**Table 28.91 CFERRINT_VCI2CFBB register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 | ADEDIE | Address ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Address ECC error detection/ correction is enabled<br>0: Address ECC 2-bit error report disabled<br>1: Address ECC 2-bit error report enabled |
| 4 | ASEDIE | Address ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Address ECC error detection/ correction is enabled<br>0: Address ECC 1-bit error report disabled<br>1: Address ECC 1-bit error report enabled |
| 3 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (3) CFERSTCLR_VCI2CFBB — FLI (Code-Flash) Address ECC SED/DED Status Clear Register

CFERSTCLR_VCI2CFBB register is used to clear error flag in CFERSTR_VCI2CFBB, error overflow flag in CFOVFSTR_VCI2CFBB, and error address in CFEADR0_VCI2CFBB. These are write only registers and read value is always “0”.

**Access:** These registers can be written only in 32/16/8-bit units.

**Address:** FFC6 2C08$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.92 CFERSTCLR_VCI2CFBB register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SSTCLR | Clear the error status flags<br>Clear ADEDF/ASEDF in CFERSTR_VCI2CFBB, ERROVF in CFOVFSTR_VCI2CFBB, and CFEADR0_VCI2CFBB writing “1” to this bit. |

### (4) CFOVFSTR_VCI2CFBB — FLI (Code-Flash) Address Error Count Overflow Status Register

CFOVFSTR_VCI2CFBB register monitors if address error overflow occurs. Overflow occurs when different error*1 is detected in the case that error status has been already set. If the error in error status is the same error (same error cause and same error address) with the errors already stored, this flag is not set. ERROVF flag is cleared by system reset or SSTCLR in CFERSTCLR_VCI2CFBB register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** FFC6 2C0C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.93 CFOVFSTR_VCI2CFBB register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ERROVF | Error overflow flag<br>If an error flag (ADEDF/ASEDF in CFERSTR_VCI2CFBB) is set and a different error is detected, this bit is set. |

### (5) CFERSTR_VCI2CFBB — FLI (Code-Flash) Address ECC SED/DED Status Register

CFERSTR_VCI2CFBB is the address error monitor register. When all error flag is “0” for each bank and a new the error occurs, error status flag is set. If address ECC 1-bit error monitor flag is set and the new error is address ECC 2 bit error, the new error is set (does not clear the previous error flag). If a multiple error causes are detected, detected errors are all set (e.g. if ADED is detected at the same access, ADEDF is set.). This register is cleared by system reset or SSTCLR in CFERSTCLR_VCI2CFBB register.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** FFC6 2C10$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ADEDF | ASEDF | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.94 CFERSTR_VCI2CFBB register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is read. |
| 7 | ADEDF | Address ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in CFERSTCLR_VCI2CFBB<br>Condition for “1”: ADEDF is “0” and ECC 2-bit error is detected. |
| 6 | ASEDF | Address ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in CFERSTCLR_VCI2CFBB<br>Condition for “1”: ASEDF is “0” and ECC 1-bit error is detected. |
| 5 to 0 | Reserved | When read, the value after reset is read. |

### (6) CFEADR0_VCI2CFBB — FLI (Code-Flash) Address ECC SED/DED Address Register

CFEADR0_VCI2CFBB register is used to hold the address when all error flags are not set and an error is detected. If ASEDF in CFERSTR_VCI2CFBB is set and address ECC 2-bit error is detected. This register stores a corrected address in case of 1-bit error when address correction is enabled. But when address correction is disabled or address has error on bits to exceed 2 bits the address with error is stored.

CFEADR0_VCI2CFBB is cleared by system reset or SSTCLR in CFERSTCLR_VCI2CFBB register.

**Access:** These registers can be read only in 32-bit units.

**Address:** FFC6 2C50$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.95 CFEADR0_VCI2CFBB register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | EADR[31:0] | Error detection address |

### (7) IFECCCTL_PE1/PE2 — IFU Data ECC Control Register

IFECCCTL_PE1/PE2 registers control the ECC error detection/correction and 1-bit error correction on master port in instruction fetch unit that is used to fetch an instruction to global or local RAM. Writing ECC control registers must be executed with PROT[1:0] = 01.

**Access:** These registers can be read/written in 32/16-bit units.

**Address:** IFECCCTL_PE1: FFC6 3400$_H$
IFECCCTL_PE2: FFC6 3600$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.96 IFECCCTL_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15 | PROT1 | Enables or disables modification of the ECCDIS and SECDIS bits. The value written is not retained. These bits are always read as 0.<br>Set (PROT1, PROT0) = (0, 1) when writing to IFECCCTL_PE1/PE2. |
| 14 | PROT0 | |
| 13 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SECDIS | 1-Bit Error Correction Disable<br>Enables or disables 1-bit error correction when ECC error detection and correction are enabled. Write a value to this bit simultaneously with setting (PROT1, PROT0) = (0, 1).<br>0: Enables correction of the 1-bit error detected.<br>1: Disables correction of the 1-bit error detected. |
| 0 | ECCDIS | ECC Disable<br>Enables or disables ECC error detection and correction. Write a value to this bit simultaneously with setting (PROT1, PROT0) = (0, 1).<br>0: Enables ECC error detection and correction.<br>1: Disables ECC error detection and correction. |

### (8) IFERRINT_PE1/PE2 — IFU Data Error Information Control Register

IFERRINT_PE1/PE2 registers control whether error information is reported to ECM, when data ECC 2-bit error and data ECC 1-bit error are detected.

**Access:** These registers can be read/written in 32/16/8-bit units.

**Address:** IFERRINT_PE1: FFC6 3404$_H$
IFERRINT_PE2: FFC6 3604$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.97 IFERRINT_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | ECC 2-bit error Notification Enable<br>Enables or disables generation of the error notification signal upon detection of a 2-bit error when ECC error detection and correction are enabled.<br>0: Disables notification of the ECC 2-bit error.<br>1: Enables notification of the ECC 2-bit error. |
| 0 | SEDIE | ECC 1-bit error Notification Enable<br>Enables or disables generation of the error notification signal upon detection of a 1-bit error when ECC error detection and correction are enabled.<br>0: Disables notification of the ECC 1-bit error.<br>1: Enables notification of the ECC 1-bit error. |

### (9) IFERSTCLR_PE1/PE2 — IFU Data ECC SED/DED Status Clear Register

IFERSTCLR_PE1/PE2 registers are used to clear SEDF and DEDF in IFERSTR_PE1/PE2 and ERROVF in IFOVFSTR_PE1/PE2, and error address in IFERADR_PE1/PE2. These are write only registers and read value is always “0”.

**Access:** These registers can be written only in 32/16/8-bit units.

**Address:** IFERSTCLR_PE1: FFC6 $3408_H$
IFERSTCLR_PE2: FFC6 $3608_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | STCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.98 IFERSTCLR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | STCLR | Clear the error status flags<br>Writing 1 to this bit clears the DEDF and SEDF flags in IFERSTR_PE1/PE2; ERROVF flag in IFOVFSTR_PE1/PE2; and IFERADR_PE1/PE2.EADR. |

### (10) IFOVFSTR_PE1/PE2 — IFU Data Error Count Overflow Status Register

IFOVFSTR_PE1/PE2 registers monitor if error overflow occurs. Overflow occurs when different error *1 is detected in the case that error status has been already set. If the error in error status is the same error (same error cause and same error address) with the errors already stored, this flag is not set. ERROVF flag is cleared by system reset or STCLR in IFERSTCLR_PE1/PE2 register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** IFOVFSTR_PE1: FFC6 340C$_H$
IFOVFSTR_PE2: FFC6 360C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.99 IFOVFSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ERROVF | Error Overflow Flag<br>ERROVF is set if the second error occurs while any of the error flags (DEDF and SEDF) in the error status register is set, except when both of the error address and source of the second error are the same as those of the first error. |

### (11) IFERSTR_PE1/PE2 — IFU Data ECC SED/DED Status Register

IFERSTR_PE1/PE2 are the error monitor registers. When all error flag is “0” and a new error occurs, error status flag is set. If data ECC 1 bit is set and the new error is data ECC 2 bit, the new error is set (does not clear the previous error flag).

IFERSTR_PE1/PE2 register is cleared by system reset or STCLR in IFERSTCLR_PE1/PE2 register.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** IFERSTR_PE1: FFC6 3410$_H$
IFERSTR_PE2: FFC6 3610$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDF | SEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.100 IFERSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF | ECC 2-bit error Monitor Flag<br>0: Cleared to 0 by setting the STCLR bit to 1 in IFERSTCLR_PE1/PE2.<br>1: Indicates that an ECC 2-bit error has occurred while the error flag DEDF is 0. |
| 0 | SEDF | ECC 1-bit error Monitor Flag<br>0: Cleared to 0 by setting the STCLR bit to 1 in IFERSTCLR_PE1/PE2.<br>1: Indicates that an ECC 1-bit error has occurred while the error flags DEDF and SEDF are 0. |

### (12) IFERADR_PE1/PE2 — IFU Data ECC SED/DED Address Register

IFERADR_PE1/PE2 registers are used to hold the address when all error flags are not set and an error is detected. If SEDF in IFERSTR_PE1/PE2 is set and data ECC 2-bit error is detected.

IFERADR_PE1/PE2 is cleared by system reset or STCLR in IFERSTCLR_PE1/PE2 register.

**Access:** These registers can be read only in 32-bit units.

**Address:** IFERADR_PE1: FFC6 3450$_H$
IFERADR_PE2: FFC6 3650$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.101 IFERADR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | EADR[31:0] | Error detection address |

### (13) LSSECCCTL_PE1/PE2 — LSU Slave Data and Address ECC Control Register

LSSECCCTL_PE1/PE2 registers control the ECC error detection/correction and 1-bit error correction on slave port in load store unit that is used to access into local RAM from other master. Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

**Access:** These registers can be read/written in 32/16-bit units.

**Address:** LSSECCCTL_PE1: FFC6 $6000_H$
LSSECCCTL_PE2: FFC6 $6200_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.102 LSSECCCTL_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SECDIS | Address and Data ECC 1-bit error correction enable bit<br>When using Address and Data ECC error detection/correction (ECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is carried out when 1-bit is detected<br>1: Correction is not carried out when 1-bit is detected |
| 0 | ECCDIS | Address and Data ECC disable bit<br>Setting Address and Data ECC error detection/correction to enable/disable.<br>0: Address and Data ECC error detection/correction is enable<br>1: Address and Data ECC error detection/correction is disable |

### (14) LSSERRINT_PE1/PE2 — LSU Slave Data and Address Error information Control Register

LSSERRINT_PE1/PE2 registers control whether error information is reported to ECM, when data/address ECC 2-bit error and data/address ECC 1-bit error are detected.

**Access:** These registers can be read/written in 32/16/8-bit units.

**Address:** LSSERRINT_PE1: FFC6 6004$_H$
LSSERRINT_PE2: FFC6 6204$_H$

**Value after reset:** 0000 0033$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ADEDIE | ASEDIE | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R/W | R/W |

**Table 28.103 LSSERRINT_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 | ADEDIE | Address ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Address ECC error detection/ correction is enabled<br>0: Address ECC 2-bit error report disabled<br>1: Address ECC 2-bit error report enabled |
| 4 | ASEDIE | Address ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Address ECC error detection/ correction is enabled<br>0: Address ECC 1-bit error report disabled<br>1: Address ECC 1-bit error report enabled |
| 3, 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | Data ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 2-bit error report disabled<br>1: Data ECC 2-bit error report enabled |
| 0 | SEDIE | Data ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 1-bit error report disabled<br>1: Data ECC 1-bit error report enabled |

### (15) LSSERSTCLR_PE1/PE2 — LSU Slave Data and Address ECC SED/DED Status Clear Register

LSSERRSTCLR_PE1/PE2 registers are used to clear error flag in LSSERSTR_PE1/PE2, error overflow flag in LSSOVFSTR_PE1/PE2, and error address in LSSEADR_PE1/PE2. These are write only registers and read value is always “0”.

**Access:** These registers can be written only in 32/16/8-bit units.

**Address:** LSSERSTCLR_PE1: FFC6 6008$_H$
LSSERSTCLR_PE2: FFC6 6208$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.104 LSSERSTCLR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SSTCLR | Clear the error status flags<br>Clear ADEDF / ASEDF / DEDF / SEDF in LSSERSTR_PE1/PE2, ERROVF in LSSOVFSTR_PE1/PE2, and LSSEADR_PE1/PE2 writing “1” to this bit. |

### (16) LSSOVFSTR_PE1/PE2 — LSU Slave Data and Address Error Count Overflow Status Register

LSSOVFSTR_PE1/PE2 registers monitor if error overflow occurs. Overflow occurs when different error*1 is detected in the case that error status has been already set. If the error in error status is the same error (same error cause and same error address) with the errors already stored, this flag is not set. ERROVF flag is cleared by system reset or SSTCLR in LSSERSTCLR_PE1/PE2 register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** LSSOVFSTR_PE1: FFC6 600C$_H$
LSSOVFSTR_PE2: FFC6 620C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.105 LSSOVFSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ERROVF | Error overflow flag<br>If an error flag (ADEDF, ASEDF, DEDF, SEDF in LSERSTR_PE1/PE2) is set and a different error is detected, this bit is set. |

### (17) LSSERSTR_PE1/PE2 — LSU Slave Data and Address ECC SED/DED Status Register

LSSERSTR_PE1/PE2 are the error monitor registers. When all error flag is “0” and a new the error occurs, error status flag is set. If address ECC 1 bit, data ECC 1 bit is set and the new error is address ECC 2 bit, data ECC 2 bit, the new error is set (does not clear the previous error flag). If a multiple error causes are detected, detected errors are all set (e.g. if DED and ADED are detected at the same access, DEDF and ADEDF are both set.). LSSERSTR_PE1/PE2 registers are cleared by system reset or SSTCLR in LSSERSTCLR_PE1/PE2 register.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** LSSERSTR_PE1: FFC6 $6010_H$
LSSERSTR_PE2: FFC6 $6210_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ADEDF | ASEDF | — | — | — | — | DEDF | SEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.106 LSSERSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is read. |
| 7 | ADEDF | Address ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in LSSERSTCLR_PE1/PE2<br>Condition to “1”: ADEDF is “0” and ECC 2-bit error is detected. |
| 6 | ASEDF | Address ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in LSSERSTCLR_PE1/PE2<br>Condition to “1”: ASEDF is “0” and ECC 1-bit error is detected. |
| 5 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF | Data ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in LSSERSTCLR_PE1/PE2<br>Condition to “1”: DEDF is “0” and ECC 2-bit error is detected. |
| 0 | SEDF | Data ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in LSSERSTCLR_PE1/PE2<br>Condition to “1”: SEDF is “0” and ECC 1-bit error is detected. |

### (18) LSSEADR_PE1/PE2 — LSU Slave Data and Address ECC SED/DED Address Register

LSSEADR_PE1/PE2 registers are used to hold the address when all error flags are not set and an error is detected. If ASEDF/SEDF in LSSERSTR_PE1/PE2 is set and address ECC 2 bit/data ECC 2-bit error is detected. This register stores a corrected address in case of 1- bit error when address correction is enabled. But when address correction is disabled or address has error on bits to exceed 2 bits the address with error is stored.

LSSEADR_PE1/PE2 are cleared by system reset or SSTCLR in LSSERSTCLR_PE1/PE2 register.

**Access:** These registers can be read only in 32-bit units.

**Address:** LSSEADR_PE1: FFC6 $6050_H$
LSSEADR_PE2: FFC6 $6250_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.107 LSSEADR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | EADR[31:0] | Error detection address |

### (19) LSMECCCTL_PE1/PE2 — LSU Master Data ECC Control Register

LSMECCCTL_PE1/PE2 registers control the ECC error detection/correction and 1-bit error correction on master port in load store unit that is used to access into data in RAM or peripheral registers. Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

**Access:** These registers can be read/written in 32/16-bit units.

**Address:** LSMECCCTL_PE1: FFC6 $6400_H$
LSMECCCTL_PE2: FFC6 $6600_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.108 LSMECCCTL_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SECDIS | Data ECC 1-bit error correction enable bit<br>When using Data ECC error detection/correction (ECCDIS = 0), setting 1-bit error<br>correction to enable/disable.<br>0: Correction is done when 1-bit error is detected<br>1: No Correction is done when 1-bit error is detected |
| 0 | ECCDIS | Data ECC disable bit<br>Setting Data ECC error detection/correction to enable/disable.<br>0: Data ECC error detection/correction is enable<br>1: Data ECC error detection/correction is disable |

### (20) LSMERRINT_PE1/PE2 — LSU Master Data Error Information Control Register

LSMERRINT_PE1/PE2 registers control whether error information is reported to ECM, when data ECC 2-bit error and data ECC 1-bit error are detected.

**Access:** These registers can be read/written in 32/16/8-bit units.

**Address:** LSMERRINT_PE1: FFC6 6404$_H$
LSMERRINT_PE2: FFC6 6604$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.109 LSMERRINT_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | Data ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 2-bit error report disabled<br>1: Data ECC 2-bit error report enabled |
| 0 | SEDIE | Data ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 1-bit error report disabled<br>1: Data ECC 1-bit error report enabled |

### (21) LSMERSTCLR_PE1/PE2 — LSU Master Data ECC SED/DED Status Clear Register

LSMERRSTCLR_PE1/PE2 registers are used to clear error flag in LSMERSTR_PE1/PE2, error overflow flag in LSMOVFSTR_PE1/PE2, and error address in LSMEADR_PE1/PE2. These are write only registers and read value is always “0”.

**Access:** These registers can be written only in 32/16/8-bit units.

**Address:** LSMERSTCLR_PE1: FFC6 6408$_H$
LSMERSTCLR_PE2: FFC6 6608$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.110 LSMERSTCLR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SSTCLR | Clear the error status flags<br>Clear DEDF / SEDF in LSMERSTR_PE1/PE2, ERROVF in LSMOVFSTR_PE1/PE2, and LSMEADR_PE1/PE2 writing “1” to this bit. |

### (22) LSMOVFSTR_PE1/PE2 — LSU Master Data Error Count Overflow Status Register

LSMOVFSTR_PE1/PE2 registers monitor if error overflow occurs. Overflow occurs when different error*1 is detected in the case that error status has been already set. If the error in error status is the same error (same error cause and same error address) with the errors already stored, this flag is not set. ERROVF flag is cleared by system reset or SSTCLR in LSMERSTCLR_PE1/PE2 register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** LSMOVFSTR_PE1: FFC6 640C$_H$
LSMOVFSTR_PE2: FFC6 660C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.111 LSMOVFSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ERROVF | Error overflow flag<br>If an error flag (DEDF, SEDF in LSMERSTR_PE1/PE2) is set and a different error is detected, this bit is set. |

### (23) LSMERSTR_PE1/PE2 — LSU Master Data ECC SED/DED Status Register

LSMERSTR_PE1/PE2 are the error monitor registers. When all error flag is “0” for each bank and a new the error occurs, error status flag is set. If data ECC 1 bit is set and the new error is data ECC 2 bit, the new error is set (does not clear the previous error flag). LSMERSTR_PE1/PE2 register is cleared by system reset or SSTCLR in LSMERSTCLR_PE1/PE2 register.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** LSMERSTR_PE1: FFC6 6410$_H$
LSMERSTR_PE2: FFC6 6610$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDF | SEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.112 LSMERSTR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF | Data ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in LSMERSTCLR_PE1/PE2<br>Condition to “1”: DEDF is “0” and ECC 2-bit error is detected. |
| 0 | SEDF | Data ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in LSMERSTCLR_PE1/PE2<br>Condition to “1”: SEDF is “0” and ECC 1-bit error is detected. |

### (24) LSMEADR_PE1/PE2 — LSU Master Data ECC SED/DED Address Register

LSMEADR_PE1/PE2 registers are used to hold the address when all error flags are not set and an error is detected. If SEDF in LSMERSTR_PE1/PE2 is set and data ECC 2-bit error is detected.

LSMEADR_PE1/PE2 is cleared by system reset or SSTCLR in LSMERSTCLR_PE1/PE2 register.

**Access:** These registers can be read only in 32-bit units.

**Address:** LSMEADR_PE1: FFC6 $6450_H$
LSMEADR_PE2: FFC6 $6650_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.113 LSMEADR_PE1/PE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | EADR[31:0] | Error detection address |

### (25) VPECCCTL_SG*n* — System Interconnect (PBus I/F) Data ECC Control Register (*n* = 0 to 3, 5)

VPECCCTL_SGn (n = 0 to 3, 5) register control the ECC error detection/correction and 1-bit error correction on ECC modules that are used only when the bit operation is executed to a PBus peripheral register. Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

**NOTE**

These registers can be written and read. However even if these registers are set, the setting doesn't affect the operation. ECC and 1 bit error correction can not be disabled.

**Access:** These registers can be read/written in 32/16-bit units.

**Address:** VPECCCTL_SG0: FFC6 $7000_H$
VPECCCTL_SG1: FFC6 $7400_H$
VPECCCTL_SG2: FFC6 $7800_H$
VPECCCTL_SG3: FFC6 $7C00_H$
VPECCCTL_SG5: FFC6 $8400_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.114 VPECCCTL_SG*n* register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SECDIS | Data ECC 1-bit error correction enable bit<br>When using Data ECC error detection/correction (ECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is done when 1-bit error is detected<br>1: No Correction is done when 1-bit error is detected |
| 0 | ECCDIS | Data ECC disable bit<br>Setting Data ECC error detection/correction to enable/disable.<br>0: Data ECC error detection/correction is enable<br>1: Data ECC error detection/correction is disable |

### (26) VPERRINT_SG*n* — System Interconnect (PBus I/F) Data Error Information Control Register (*n* = 0 to 3, 5)

VPERRINT_SG*n* (*n* = 0 to 3, 5) registers control whether error information is reported to ECM, when data ECC 2-bit error and data ECC 1-bit error are detected.

**Access:** These registers can be read/written in 32/16/8-bit units.

**Address:** VPERRINT_SG0: FFC6 7004$_H$
VPERRINT_SG1: FFC6 7404$_H$
VPERRINT_SG2: FFC6 7804$_H$
VPERRINT_SG3: FFC6 7C04$_H$
VPERRINT_SG5: FFC6 8404$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.115 VPERRINT_SG*n* register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | Data ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 2-bit error report disabled<br>1: Data ECC 2-bit error report enabled |
| 0 | SEDIE | Data ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 1-bit error report disabled<br>1: Data ECC 1-bit error report enabled |

### (27) VPERSTCLR_SG*n* — System Interconnect (PBus I/F) Data ECC SED/DED Status Clear Register (*n* = 0 to 3, 5)

VPERRSTCLR_SG*n* registers are used to clear error flag in VPERSTR_SG*n*, error overflow flag in VPOVFSTR_SGn, and error address in VPEADR_SG*n*. These are write only registers and read value is always “0”.

**Access:** These registers can be written only in 32/16/8-bit units.

**Address:** VPERSTCLR_SG0: FFC6 7008$_H$
VPERSTCLR_SG1: FFC6 7408$_H$
VPERSTCLR_SG2: FFC6 7808$_H$
VPERSTCLR_SG3: FFC6 7C08$_H$
VPERSTCLR_SG5: FFC6 8408$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.116 VPERSTCLR_SG*n* register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SSTCLR | Clear the error status flags<br>Clear DEDF / SEDF in VPERSTR_SGn, ERROVF in VPOVFSTR_SGn, and VPEADR_SGn writing “1” to this bit. |

### (28) VPOVFSTR_SG*n* — System Interconnect (PBus I/F) Data Error Count Overflow Status Register (*n* = 0 to 3, 5)

VPOVFSTR_SG*n* registers monitor if error overflow occurs. Overflow occurs when different error*1 is detected in the case that error status has been already set. If the error in error status is the same error (same error cause and same error address) with the errors already stored, this flag is not set. ERROVF flag is cleared by system reset or SSTCLR in VPERSTCLR_SG*n* register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** VPOVFSTR_SG0: FFC6 700C$_H$
VPOVFSTR_SG1: FFC6 740C$_H$
VPOVFSTR_SG2: FFC6 780C$_H$
VPOVFSTR_SG3: FFC6 7C0C$_H$
VPOVFSTR_SG5: FFC6 840C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.117 VPOVFSTR_SG*n* register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ERROVF | Error overflow flag<br>If an error flag (DEDF, SEDF in VPERSTR_SGn) is set and a different error is detected, this bit is set. |

### (29) VPERSTR_SGn — System Interconnect (PBus I/F) Data ECC Status Register (*n* = 0 to 3, 5)

VPERSTR_SG*n* is the error monitor register. If the error flag is “0” for each bank and a new error occurs, the error status flag is set. If address ECC 1 bit, data ECC 1 bit is set and the new error is address ECC 2 bit, data ECC 2 bit, the new error is set (does not clear the previous error flag). VPERSTR_SG*n* register is cleared by system reset or SSTCLR in VPERSTCLR_SG*n* register.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** VPERSTR_SG0: FFC6 $7010_H$
VPERSTR_SG1: FFC6 $7410_H$
VPERSTR_SG2: FFC6 $7810_H$
VPERSTR_SG3: FFC6 $7C10_H$
VPERSTR_SG5: FFC6 $8410_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDF | SEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.118 VPERSTR_SG*n* register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF | Data ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in VPERSTCLR_SGn<br>Condition to “1”: DEDF is “0” and ECC 2-bit error is detected. |
| 0 | SEDF | Data ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in VPERSTCLR_SGn<br>Condition to “1”: SEDF is “0” and ECC 1-bit error is detected |

### (30) VPEADR_SG*n* — System Interconnect (PBus I/F) Data ECC SED/DED Address Register (*n* = 0 to 3, 5)

VPEADR_SGn register is used to hold the address when all error flags are not set and an error is detected. If SEDF in VPERSTR_SGn is set and data ECC 2-bit error is detected.

VPEADR_SGn is cleared by system reset or SSTCLR in VPERSTCLR_SGn register.

**Access:** These registers can be read only in 32-bit units.

**Address:** VPEADR_SG0: FFC6 7050$_H$
VPEADR_SG1: FFC6 7450$_H$
VPEADR_SG2: FFC6 7850$_H$
VPEADR_SG3: FFC6 7C50$_H$
VPEADR_SG5: FFC6 8450$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.119 VPEADR_SG*n* register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | EADR[31:0] | Error detection address |

### (31) VPECCCTL_TERM_SG7 — PBus Data and Address ECC Control Register

VPECCCTL_TERM_SG7 register control the ECC error detection/correction and 1-bit error correction on the terminal of peripheral bus. There is no available peripheral for the area covered by this ECC module. Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

**NOTE**

This register can be written and read. However even if this register is set, the setting doesn't affect the operation. ECC and 1 bit error correction can not be disabled.

**Access:** This register can be read/written in 32/16-bit units.

**Address:** VPECCCTL_TERM_SG7: FFC6 8E00$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | ASECDIS | AECCDIS | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 28.120 VPECCCTL_TERM_SG7 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 | ASECDIS | Address ECC 1-bit error correction enable bit<br>When using Address ECC error detection/correction (AECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is done when 1-bit error is detected<br>1: No Correction is done when 1-bit error is detected |
| 2 | AECCDIS | Address ECC disable bit<br>Setting Address ECC error detection/correction to enable/disable.<br>0: Address ECC error detection/correction is enable<br>1: Address ECC error detection/correction is disable |
| 1 | SECDIS | Data ECC 1-bit error correction enable bit<br>When using Data ECC error detection/correction (ECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is done when 1-bit error is detected<br>1: No Correction is done when 1-bit error is detected |
| 0 | ECCDIS | Data ECC disable bit<br>Setting Data ECC error detection/correction to enable/disable.<br>0: Data ECC error detection/correction is enable<br>1: Data ECC error detection/correction is disable |

### (32) VPERRINT_TERM_SG7 — System Interconnect (PBus I/F) Data and Address Error Information Control Register

VPERRINT_TERM_SG7 register controls whether error information is reported to ECM, when data/address ECC 2-bit error and data/address ECC 1-bit error are detected.

**Access:** This register can be read/written in 32/16/8-bit units.

**Address:** VPERRINT_TERM_SG7: FFC6 8E04$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ADEDIE | ASEDIE | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R/W | R/W |

**Table 28.121 VPERRINT_TERM_SG7 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 | ADEDIE | Address ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Address ECC error detection/correction is enabled<br>0: Address ECC 2-bit error report disabled<br>1: Address ECC 2-bit error report enabled |
| 4 | ASEDIE | Address ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Address ECC error detection/correction is enabled<br>0: Address ECC 1-bit error report disabled<br>1: Address ECC 1-bit error report enabled |
| 3, 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | Data ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Data ECC error detection/correction is enabled<br>0: Data ECC 2-bit error report disabled<br>1: Data ECC 2-bit error report enabled |
| 0 | SEDIE | Data ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Data ECC error detection/correction is enabled<br>0: Data ECC 1-bit error report disabled<br>1: Data ECC 1-bit error report enabled |

### (33) VPERSTCLR_TERM_SG7 — System Interconnect (PBus I/F) Data and Address ECC SED/DED Status Clear Register

VPERSTCLR_TERM_SG7 register is used to clear error flag in VPERSTR_TERM_SG7, error overflow flag in VPOVFSTR_TERM_SG7, and error address in VPEADR_TERM_SG7. This is write only register and read value is always “0”.

**Access:** This register can be written only in 32/16/8-bit units.

**Address:** VPERSTCLR_TERM_SG7: FFC6 8E08$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.122 VPERSTCLR_TERM_SG7 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SSTCLR | Clear the error status flags<br>Clear ADEDF / ASEDF / DEDF / SEDF in VPERSTR_TERM_SG7, ERROVF in VPOVFSTR_TERM_SG7, and VPEADR_TERM_SG7 writing “1” to this bit. |

### (34) VPOVFSTR_TERM_SG7 — System Interconnect (PBus I/F) Data Error Count Overflow Status Register

VPOVFSTR_TERM_SG7 register monitor if error overflow occurs. Overflow occurs when different error*1 is detected in the case that error status has been already set. If the error in error status is the same error (same error cause and same error address) with the errors already stored, this flag is not set. ERROVF flag is cleared by system reset or SSTCLR in VPERSTCLR_TERM_SG7 register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** This register can be read only in 32/16/8-bit units.

**Address:** VPOVFSTR_TERM_SG7: FFC6 8E0C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.123 VPOVFSTR_TERM_SG7 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ERROVF | Error overflow flag<br>If an error flag (ADEDF, ASEDF, DEDF, SEDF in VPERSTR_TERM_SG7) is set and a different error is detected, this bit is set. |

### (35) VPERSTR_TERM_SG7 — System Interconnect (PBus I/F) Data and Address ECC SED/DED Status Register

VPERSTR_TERM_SG7 is the error monitor register. If the error flag is “0” for each bank and a new error occurs, the error status flag is set. If address ECC 1 bit, data ECC 1 bit is set and the new error is address ECC 2 bit, data ECC 2 bit, the new error is set (does not clear the previous error flag). If a multiple error causes are detected, detected errors are all set (e.g. if DED and ADED are detected at the same access, DEDF and ADEDF are both set.). VPERSTR_TERM_SG7 register is cleared by system reset or SSTCLR in VPERSTCLR_TERM_SG7 register.

**Access:** This register can be read only in 32/16/8-bit units.

**Address:** VPERSTR_TERM_SG7: FFC6 8E10$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ADEDF | ASEDF | — | — | — | — | DEDF | SEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.124 VPERSTR_TERM_SG7 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is read. |
| 7 | ADEDF | Address ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in VPERSTCLR_TERM_SG7<br>Condition to “1”: ADEDF is “0” and ECC 2-bit error is detected. |
| 6 | ASEDF | Address ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in VPERSTCLR_TERM_SG7<br>Condition to “1”: ASEDF is “0” and ECC 1-bit error is detected. |
| 5 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF | Data ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in VPERSTCLR_TERM_SG7<br>Condition to “1”: DEDF is “0” and ECC 2-bit error is detected. |
| 0 | SEDF | Data ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in VPERSTCLR_TERM_SG7<br>Condition to “1”: SEDF is “0” and ECC 1-bit error is detected. |

### (36) VPEADR_TERM_SG7 — System Interconnect (PBus I/F) Data ECC SED/DED Address Register

VPEADR_TERM_SG7 register is used to hold the address when all error flags are not set and an error is detected. If ASEDF/SEDF in VPERSTR_TERM_SG7 is set and address ECC 2 bit/data ECC 2-bit error is detected. This register stores a corrected address in case of 1- bit error when address correction is enabled. But when address correction is disabled or address has error on bits to exceed 2 bits the address with error is stored.

VPEADR_TERM_SG7 is cleared by system reset or SSTCLR in VPERSTCLR_TERM_SG7 register.

**Access:** This register can be read only in 32-bit units.

**Address:** VPEADR_TERM_SG7: FFC6 8E50$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.125 VPEADR_TERM_SG7 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | EADR[31:0] | Error detection address |

### (37) VPECCCTL_VCI2VPI — System Interconnect Address ECC Control Register

VPECCCTL_VCI2VPI register control the ECC error detection/correction and 1-bit error correction on bus width conversion between system interconnection and PBus that is used only when data to exceed 64 bits is transferred to PBus peripheral from DMA. Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

**Access:** These registers can be read/written in 32/16-bit units.

**Address:** VPECCCTL_VCI2VPI: FFC6 $9000_H$

**Value after reset:** 0000 $0000_H$

**NOTE**

This register can be written and read. However even if this register is set, the setting doesn't affect the operation. ECC and 1 bit error correction can not be disabled.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | ASECDIS | AECCDIS | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R |

**Table 28.126 VPECCCTL_VCI2VPI register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 | ASECDIS | Address ECC 1-bit error correction enable bit<br>When using Address ECC error detection/correction (AECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is done when 1-bit error is detected<br>1: No Correction is done when 1-bit error is detected |
| 2 | AECCDIS | Address ECC disable bit<br>Setting Address ECC error detection/correction to enable/disable.<br>0: Address ECC error detection/correction is enable<br>1: Address ECC error detection/correction is disable |
| 1, 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (38) VPERRINT_VCI2VPI — System Interconnect Address ECC Error Information Control Register

VPERRINT_VCI2VPI registers control whether error information is reported to ECM, when address ECC 2-bit error and address ECC 1-bit error are detected.

**Access:** These registers can be read/written in 32/16/8-bit units.

**Address:** VPERRINT_VCI2VPI: FFC6 $9004_H$

**Value after reset:** 0000 $0003_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ADEDIE | ASEDIE | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R | R |

**Table 28.127 VPERRINT_VCI2VPI register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 | ADEDIE | Address ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Address ECC error detection/ correction is enabled<br>0: Address ECC 2-bit error report disabled<br>1: Address ECC 2-bit error report enabled |
| 4 | ASEDIE | Address ECC 1bit error report enable bit<br>Control error report of 1bit error detection when Address ECC error detection/ correction is enabled<br>0: Address ECC 1bit error report disabled<br>1: Address ECC 1bit error report enabled |
| 3 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (39) VPERSTCLR_VCI2VPI — System Interconnect Address ECC SED/DED Status Clear Register

VPERRSTCLR_VCI2VPI registers are used to clear error flag in VPERSTR_VCI2VPI, error overflow flag in VPOVFSTR_VCI2VPI, and error address in VPERADR0_VCI2VPI. These are write only registers and read value is always “0”.

**Access:** These registers can be written only in 32/16/8-bit units.

**Address:** VPERSTCLR_VCI2VPI: FFC6 $9008_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.128 VPERSTCLR_VCI2VPI register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SSTCLR | Clear the error status flags<br>Clear ADEDF / ASEDF in VPERSTR_VCI2VPI, ERROVF in VPOVFSTR_VCI2VPI, and VPEADR_VCI2VPI writing “1” to this bit. |

### (40) VPOVFSTR_VCI2VPI —System Interconnect Address Error Count Overflow Status Register

VPOVFSTR_VCI2VPI registers monitor if error overflow occurs. Overflow occurs when different error*1 is detected in the case that error status has been already set. If the error in error status is the same error (same error cause and same error address) with the errors already stored, this flag is not set. ERROVF flag is cleared by system reset or SSTCLR in VPERSTCLR_VCI2VPI register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** VPOVFSTR_VCI2VPI: FFC6 900C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.129 VPOVFSTR_VCI2VPI register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ERROVF | Error overflow flag<br>If an error flag (ADEDF, ASEDF in VPERSTR_VCI2VPI) is set and a different error is detected, this bit is set. |

### (41) VPERSTR_VCI2VPI — System Interconnect Address ECC Status Register

VPERSTR_VCI2VPI is the error monitor register. If the error flag is “0” for each bank and a new error occurs, the error status flag is set. If address ECC 1 bit is set and the new error is address ECC 2 bit, the new error is set (does not clear the previous error flag). VPERSTR_VCI2VPI register is cleared by system reset or SSTCLR in VPERSTCLR_VCI2VPI register.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** VPERSTR_VCI2VPI: FFC6 $9010_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ADEDF | ASEDF | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.130 VPERSTR_VCI2VPI register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is read. |
| 7 | ADEDF | Address ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in VPERSTCLR_VCI2VPI<br>Condition to “1”: ADEDF is “0” and ECC 2-bit error is detected. |
| 6 | ASEDF | Address ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in VPERSTCLR_VCI2VPI<br>Condition to “1”: ASEDF is “0” and ECC 1-bit error is detected |
| 5 to 0 | Reserved | When read, the value after reset is read. |

### (42) VPERADR0_VCI2VPI — System Interconnect Address ECC SED/DED Address Register

VPERADR0_VCI2VPI register is used to hold the address when all error flags are not set and an error is detected. If SEDF in VPERSTR_VCI2VPI is set and data ECC 2-bit error is detected.

VPERADR0_VCI2VPI is cleared by system reset or SSTCLR in VPERSTCLR_VCI2VPI register.

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** VPERADR0_VCI2VPI: FFC6 9050$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.131 VPERADR0_VCI2VPI register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | EADR[31:0] | Error detection address |

### (43) VCECCCTL_PDMA — System Interconnect Data ECC Control Register

VCECCCTL_PDMA register control the ECC error detection/correction and 1-bit error correction on master port of DMA that is used to read to a source from DMA. Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

**Access:** This register can be read/written in 32/16-bit units.

**Address:** FFC6 $A000_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | — | — | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.132 VCECCCTL_PDMA register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | SECDIS | Data ECC 1-bit error correction enable bit<br>When using Data ECC error detection/correction (ECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is done when 1-bit error is detected<br>1: No Correction is done when 1-bit error is detected |
| 0 | ECCDIS | Data ECC disable bit<br>Setting Data ECC error detection/correction to enable/disable.<br>0: Data ECC error detection/correction is enable<br>1: Data ECC error detection/correction is disable |

### (44) VCERRINT_PDMA — System Interconnect Data Error Information Control Register

VCERRINT_PDMA register controls whether error information is reported to ECM, when data ECC 2-bit error and data ECC 1-bit error are detected.

**Access:** This register can be read/written in 32/16/8-bit units.

**Address:** FFC6 A004$_H$

**Value after reset:** 0000 0003$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.133 VCERRINT_PDMA register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | Data ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 2-bit error report disabled<br>1: Data ECC 2-bit error report enabled |
| 0 | SEDIE | Data ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 1-bit error report disabled<br>1: Data ECC 1-bit error report enabled |

### (45) VCERSTCLR_PDMA — System Interconnect Data ECC SED/DED Status Clear Register

VCERSTCLR_PDMA register is used to clear error flag in VCERSTR_PDMA, error overflow flag in VCOVFSTR_PDMA, and error address in VCEADR_PDMA. This is write only register and read value is always “0”.

**Access:** This register can be written only in 32/16/8-bit units.

**Address:** FFC6 A008$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.134 VCERSTCLR_PDMA register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SSTCLR | Clear the error status flags<br>Clear DEDF / SEDF in VCERSTR_PDMA, ERROVF in VCOVFSTR_PDMA, and VCEADR_PDMA writing “1” to this bit. |

### (46) VCOVFSTR_PDMA — System Interconnect Data Error Count Overflow Status Register

VCOVFSTR_PDMA register monitor if error overflow occurs. Overflow occurs when different error *1 is detected in the case that error status has been already set. If the error in error status is the same error (same error cause and same error address) with the errors already stored, this flag is not set. ERROVF flag is cleared by system reset or SSTCLR in VCERSTCLR_PDMA register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** This register can be read only in 32/16/8-bit units.

**Address:** FFC6 A00C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.135 VCOVFSTR_PDMA register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ERROVF | Error overflow flag<br>If an error flag (DEDF, SEDF in VCERSTR_PDMA) is set and a different error is detected, this bit is set. |

### (47) VCERSTR_PDMA — System Interconnect Data ECC SED/DED Status Register

VCERSTR_PDMA is the error monitor register. If the error flag is “0” and a new error occurs, the error status flag is set. If data ECC 1-bit error monitor flag is set and the new error is data ECC 2-bit error, the new error is set (does not clear the previous error flag). VCERSTR_PDMA register is cleared by system reset or SSTCLR in VCERSTCLR_PDMA register.

**Access:** This register can be read only in 32/16/8-bit units.

**Address:** FFC6 A010$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DEDF | SEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.136 VCERSTR_PDMA register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF | Data ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in VCERSTCLR_PDMA<br>Condition to “1”: DEDF is “0” and ECC 2-bit error is detected. |
| 0 | SEDF | Data ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in VCERSTCLR_PDMA<br>Condition to “1”: SEDF is “0” and ECC 1-bit error is detected. |

### (48) VCEADR_PDMA — System Interconnect Data ECC SED/DED Address Register

VCEADR_PDMA register is used to hold the address when all error flags are not set and an error is detected. If SEDF in VCERSTR_PDMA is set and data ECC 2-bit error is detected. VCEADR_PDMA is cleared by system reset or SSTCLR in VCERSTCLR_PDMA register.

**Access:** This register can be read only in 32-bit units.

**Address:** FFC6 A050$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.137 VCEADR_PDMA register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | EADR[31:0] | Error detection address |

### (49) APECCCTL_PFSS — P-Bus Data and Address ECC Control Register

APECCCTL_PFSS register control the ECC error detection/correction and 1-bit error correction on peripheral group 5. Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

**Access:** This register can be read/written in 32/16-bit units.

**Address:** FFC6 $C000_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | ASECDIS | AECCDIS | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 28.138 APECCCTL_PFSS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 | ASECDIS | Address ECC 1-bit error correction enable bit<br>When using Address ECC error detection/correction (AECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is carried out when 1-bit is detected<br>1: Correction is not carried out when 1-bit is detected |
| 2 | AECCDIS | Address ECC disable bit<br>Setting Address ECC error detection/correction to enable/disable.<br>0: Address ECC error detection/correction is enable<br>1: Address ECC error detection/correction is disable |
| 1 | SECDIS | Data ECC 1-bit error correction enable bit<br>When using Data ECC error detection/correction (ECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is carried out when 1-bit is detected<br>1: Correction is not carried out when 1-bit is detected |
| 0 | ECCDIS | Data ECC disable bit<br>Setting Data ECC error detection/correction to enable/disable.<br>0: Data ECC error detection/correction is enable<br>1: Data ECC error detection/correction is disable |

### (50) APERRINT_PFSS — P-Bus Data and Addres Error Information Control Register

APERRINT_PFSS register controls whether error information is reported to ECM, when data/address ECC 2-bit error and data/address ECC 1-bit error are detected.

**Access:** This register can be read/written in 32/16/8-bit units.

**Address:** FFC6 C004$_H$

**Value after reset:** 0000 0073$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ADEDIE | ASEDIE | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R/W | R/W |

**Table 28.139 APERRINT_PFSS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 | ADEDIE | Address ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Address ECC error detection/ correction is enabled<br>0: Address ECC 2-bit error report disabled<br>1: Address ECC 2-bit error report enabled |
| 4 | ASEDIE | Address ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Address ECC error detection/ correction is enabled<br>0: Address ECC 1-bit error report disabled<br>1: Address ECC 1-bit error report enabled |
| 3, 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | Data ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 2-bit error report disabled<br>1: Data ECC 2-bit error report enabled |
| 0 | SEDIE | Data ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 1-bit error report disabled<br>1: Data ECC 1-bit error report enabled |

### (51) APERSTCLR_PFSS — P-Bus Data and Address ECC SED/DED Status Clear Register

APERSTCLR_PFSS register is used to clear error flag in APERSTR_PFSS, error overflow flag in APOVFSTR_PFSS, and error address in APEADR_PFSS. This is write only register and read value is always “0”.

**Access:** This register can be written only in 32/16/8-bit units.

**Address:** FFC6 C008$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.140 APERSTCLR_PFSS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SSTCLR | Clear the error status flags<br>Clear ADEDF / ASEDF / DEDF / SEDF in APERSTR_PFSS, ERROVF in APOVFSTR_PFSS, and APEADR_PFSS writing “1” to this bit. |

### (52) APOVFSTR_PFSS — P-Bus Data and Address Error Count Overflow Status Register

APOVFSTR_PFSS register monitor if error overflow occurs. Overflow occurs when different error*1 is detected in the case that error status has been already set. If the error in error status is the same error (same error cause and same error address) with the errors already stored, this flag is not set. ERROVF flag is cleared by system reset or SSTCLR in APERSTCLR_PFSS register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** This register can be read only in 32/16/8-bit units.

**Address:** FFC6 C00C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.141 APOVFSTR_PFSS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ERROVF | Error overflow flag<br>If an error flag (ADEDF, ASEDF, DEDF, SEDF in APERSTR_PFSS) is set and a different error is detected, this bit is set. |

### (53) APERSTR_PFSS — P-Bus Data and Address ECC SED/DED Status Register

APERSTR_PFSS is the error monitor register. If the error flag is “0” for each bank and a new error occurs, the error status flag is set. If address ECC 1 bit, data ECC 1 bit is set and the new error is address ECC 2 bit, data ECC 2 bit, the new error is set (does not clear the previous error flag). If a multiple error causes are detected, detected errors are all set (e.g. if DED and ADED are detected at the same access, DEDF and ADEDF are both set.). APERSTR_PFSS register is cleared by system reset or SSTCLR in APERSTCLR_PFSS register.

**Access:** This register can be read only in 32/16/8-bit units.

**Address:** FFC6 C010$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ADEDF | ASEDF | — | — | — | — | DEDF | SEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.142 APERSTR_PFSS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is read. |
| 7 | ADEDF | Address ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in APERSTCLR_PFSS<br>Condition to “1”: ADEDF is “0” and ECC 2-bit error is detected. |
| 6 | ASEDF | Address ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in APERSTCLR_PFSS<br>Condition to “1”: ASEDF is “0” and ECC 1-bit error is detected. |
| 5 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF | Data ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in APERSTCLR_PFSS<br>Condition to “1”: DEDF is “0” and ECC 2-bit error is detected. |
| 0 | SEDF | Data ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in APERSTCLR_PFSS<br>Condition to “1”: SEDF is “0” and ECC 1-bit error is detected. |

### (54) APEADR_PFSS — P-Bus Data and Address ECC SED/DED Address Register

APEADR_PFSS register is used to hold the address when all error flags are not set and an error is detected. If ASEDF/SEDF in APERSTR_PFSS is set and address ECC 2 bit/data ECC 2-bit error is detected. This register stores a corrected address in case of 1- bit error when address correction is enabled. But when address correction is disabled or address has error on bits to exceed 2 bits the address with error is stored.

APEADR_PFSS is cleared by system reset or SSTCLR in APERSTCLR_PFSS register.

**Access:** This register can be read only in 32-bit units.

**Address:** FFC6 C050$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.143 APEADR_PFSS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | EADR[31:0] | Error detection address |

### (55) APECnECCCTL — P-Bus Data and Address ECC Control Register

APECnECCCTL (n = 0 to 6) register control the ECC error detection/correction and 1-bit error correction on peripheral group 1 to 4 (see **Table 28.145**). Writing ECC control registers must be executed with PROT[1:0] = $01_B$.

**Access:** This register can be read/written in 32/16-bit units.

**Address:** APEC0ECCCTL: FFCB $8000_H$ APEC1ECCCTL: FFE8 $8000_H$ APEC2ECCCTL: FFE8 $8200_H$
APEC3ECCCTL: FFF9 $8000_H$ APEC4ECCCTL: FFCD $8000_H$ APEC5ECCCTL: FFCD $8200_H$
APEC6ECCCTL: FFCD $8400_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | — | — | — | — | — | — | — | — | — | — | ASECDIS | AECCDIS | SECDIS | ECCDIS |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 28.144 APECnECCCTL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 15, 14 | PROT[1:0] | Protection Bit<br>01: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 13 to 4 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 3 | ASECDIS | Address ECC 1-bit error correction enable bit<br>When using Address ECC error detection/correction (AECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is carried out when 1-bit is detected<br>1: Correction is not carried out when 1-bit is detected |
| 2 | AECCDIS | Address ECC disable bit<br>Setting Address ECC error detection/correction to enable/disable.<br>0: Address ECC error detection/correction is enable<br>1: Address ECC error detection/correction is disable |
| 1 | SECDIS | Data ECC 1-bit error correction enable bit<br>When using Data ECC error detection/correction (ECCDIS = 0), setting 1-bit error correction to enable/disable.<br>0: Correction is carried out when 1-bit is detected<br>1: Correction is not carried out when 1-bit is detected |
| 0 | ECCDIS | Data ECC disable bit<br>Setting Data ECC error detection/correction to enable/disable.<br>0: Data ECC error detection/correction is enable<br>1: Data ECC error detection/correction is disable |

**Table 28.145 Target peripherals of APECnxxxx**

| ECC module | target peripheral |
|---|---|
| APEC0xxxx | Peripheral group 1 (except Aurora) |
| APEC1xxxx | Peripheral group 2 (except GTM) |
| APEC2xxxx | GTM |
| APEC3xxxx | Peripheral group 3 |
| APEC4xxxx | Peripheral group 4 (except ADCFn) |
| APEC5xxxx | ADCF0 |
| APEC6xxxx | ADCF1 |

### (56) APECnERRINT — P-Bus Data and Address Error Information Control Register

APECnERRINT (n = 0 to 6) register controls whether error information is reported to ECM, when address/data ECC 2-bit error, address/data ECC 1bit error, and ECC error overflow are detected.

**Access:** This register can be read/written in 32/16/8-bit units.

**Address:** APEC0ERRINT: FFCB 8004$_H$ APEC1ERRINT: FFE8 8004$_H$ APEC2ERRINT: FFE8 8204$_H$
APEC3ERRINT: FFF9 8004$_H$ APEC4ERRINT: FFCD 8004$_H$ APEC5ERRINT: FFCD 8204$_H$
APEC6ERRINT: FFCD 8404$_H$

**Value after reset:** 0000 0073$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ADEDIE | ASEDIE | — | — | DEDIE | SEDIE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R/W | R/W |

**Table 28.146 APECnERRINT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5 | ADEDIE | Address ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Address ECC error detection/ correction is enabled<br>0: Address ECC 2-bit error report disabled<br>1: Address ECC 2-bit error report enabled |
| 4 | ASEDIE | Address ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Address ECC error detection/ correction is enabled<br>0: Address ECC 1-bit error report disabled<br>1: Address ECC 1-bit error report enabled |
| 3, 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | DEDIE | Data ECC 2-bit error report enable bit<br>Control error report of 2-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 2-bit error report disabled<br>1: Data ECC 2-bit error report enabled |
| 0 | SEDIE | Data ECC 1-bit error report enable bit<br>Control error report of 1-bit error detection when Data ECC error detection/ correction is enabled<br>0: Data ECC 1-bit error report disabled<br>1: Data ECC 1-bit error report enabled |

### (57) APECnSTCLR — P-Bus Data and Address ECC SED/DED Status Clear Register

APECnSTCLR (n = 0 to 6) register is used to clear error flag in APECn1STERSTR, error overflow flag in APECnOVFSTR, and error address in APECn1STEADR0. This is write only register and read value is always “0”.

**Access:** This register can be written only in 32/16/8-bit units.

**Address:** APEC0STCLR: FFCB 8008$_H$ APEC1STCLR: FFE8 8008$_H$ APEC2STCLR: FFE8 8208$_H$
APEC3STCLR: FFF9 8008$_H$ APEC4STCLR: FFCD 8008$_H$ APEC5STCLR: FFCD 8208$_H$
APEC6STCLR: FFCD 8408$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTCLR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 28.147 APECnSTCLR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | SSTCLR | Clear the error status flags<br>Clear ADEDF / ASEDF / DEDF / SEDF in APECn1STERSTR, ERROVF in APECnOVFSTR, and APECn1STEADR0 writing “1” to this bit. |

### (58) APECnOVFSTR — P-Bus Data and Address Error Count Overflow Status Register

APECnOVFSTR (n = 0 to 6) register monitor if error overflow occurs. Overflow occurs when different error*1 is detected in the case that error status has been already set. If the error in error status is the same error (same error cause and same error address) with the errors already stored, this flag is not set. ERROVF flag is cleared by system reset or SSTCLR in APECnSTCLR register.

**Note 1.** Different means that the error satisfying one of the following conditions.

- Detected address is different
- Detected error status flag is different.

**Access:** This register can be read only in 32/16/8-bit units.

**Address:** APEC0OVFSTR: FFCB 800C$_H$ APEC1OVFSTR: FFE8 800C$_H$ APEC2OVFSTR: FFE8 820C$_H$
APEC3OVFSTR: FFF9 800C$_H$ APEC4OVFSTR: FFCD 800C$_H$ APEC5OVFSTR: FFCD 820C$_H$
APEC6OVFSTR: FFCD 840C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERROVF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.148 APECnOVFSTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. |
| 0 | ERROVF | Error overflow flag<br>If an error flag (ADEDF, ASEDF, DEDF, SEDF in APECn1STERSTR) is set and a different error is detected, this bit is set. |

### (59) APECn1STERSTR — P-Bus Data and Address ECC SED/DED Status Register

APECn1STERSTR (n = 0 to 6) is the error monitor register. If the error flag is “0” for each bank and a new error occurs, the error status flag is set. If address ECC 1-bit error monitor flag or data ECC 1-bit error monitor flag is set and the new error is address ECC 2-bit error or data ECC 2-bit error, the new error is set (does not clear the previous error flag). If a multiple error causes are detected, detected errors are all set (e.g. if DED and ADED are detected at the same access, DEDF and ADEDF are both set.). APECn1STERSTR register is cleared by system reset or SSTCLR in APECnSTCLR register.

**Access:** This register can be read only in 32/16/8-bit units.

**Address:** APEC01STERSTR: FFCB 8010$_H$ APEC11STERSTR: FFE8 8010$_H$ APEC21STERSTR: FFE8 8210$_H$
APEC31STERSTR: FFF9 8010$_H$ APEC41STERSTR: FFCD 8010$_H$ APEC51STERSTR: FFCD 8210$_H$
APEC61STERSTR: FFCD 8410$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ADEDF | ASEDF | — | — | — | — | DEDF | SEDF |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.149 APECn1STERSTR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When read, the value after reset is read. |
| 7 | ADEDF | Address ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in APECnSTCLR<br>Condition to “1”: ADEDF is “0” and ECC 2-bit error is detected. |
| 6 | ASEDF | Address ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in APECnSTCLR<br>Condition to “1”: ASEDF is “0” and ECC 1-bit error is detected. |
| 5 to 2 | Reserved | When read, the value after reset is read. |
| 1 | DEDF | Data ECC 2-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in APECnSTCLR<br>Condition to “1”: DEDF is “0” and ECC 2-bit error is detected. |
| 0 | SEDF | Data ECC 1-bit error monitor flag<br>Condition to “0”: system reset or write “1” to SSTCLR in APECnSTCLR<br>Condition to “1”: SEDF is “0” and ECC 1-bit error is detected. |

### (60) APECn1STEADR0 — P-Bus Data and Address ECC SED/DED Address Register

APECn1STEADR0 (n = 0 to 6) register is used to hold the address when all error flags are not set and an error is detected. If ASEDF/SEDF in APECn1STERSTR is set and address ECC 2 bit/data ECC 2-bit error is detected. This register stores a corrected address in case of 1- bit error when address correction is enabled. But when address correction is disabled or address has error on bits to exceed 2 bits the address with error is stored.

APECn1STEADR0 is cleared by system reset or SSTCLR in APECnSTCLR register.

**Access:** This register can be read only in 32/16/8-bit units.

**Address:** APEC01STEADR0: FFCB 8050$_H$ APEC11STEADR0: FFE8 8050$_H$ APEC21STEADR0: FFE8 8250$_H$
APEC31STEADR0: FFF9 8050$_H$ APEC41STEADR0: FFCD 8050$_H$ APEC51STEADR0: FFCD 8250$_H$
APEC61STEADR0: FFCD 8450$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EADR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.150 APECn1STEADR0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | EADR[31:0] | Error detection address |

#### 28.2.9.6 Test Function

The self-diagnosis operation is different by the target ECC decoder.

Error injection is not supported for VPxxxx_SGn that is used only when the bit operation is executed.

**(1) Self-diagnosis of the Data ECC check function (Slave)**

(a) Prepare data with an error to check a data ECC decoder on bus in global RAM (see **28.2.5.4** about how to set an error data into global RAM).

(b) Set ECCDIS bit in GRECCCTL_VCI2GRAM to 1 to transfer data with error into ECC decoder on bus. Also Set ECCDIS bit in VCECCCTL_PDMA to 1 to pass through DMA ECC decoder on bus if DMA ECC decoder is out of target of self-diagnosis.

(c) For a channel of DMA, Set up an address of error data in global RAM in a source address register and an address of area covered by target ECC decoder in the destination address register (For the detail of channel setup, see **Section 7, DMA**).

After that, self-diagnosis of the data ECC decoder is possible by setting software DMA transfer request flag of the channel.

This operation can be used for the following ECC decoders:

- LSU Slave data ECC: LSSxxxx of module BECCPE1 and BECCPE2 (local RAM)
- P-Bus data ECC: module BECCPBAm (a peripheral on each P-Bus group)
- DMA Data ECC on Bus: xxxx_PDMA of module BECCSIC

A word in ( ) indicates the area that should be appointed as the destination.

**(2) Self-diagnosis of the Data ECC check function (Master)**

(a) Prepare data with an error to check a data ECC decoder on bus in global RAM (see **28.2.5.4** about how to set an error data into global RAM).

(b) Set ECCDIS bit in GRECCCTL_GRAMC to transfer data with error into ECC decoder on bus.

After that, self-diagnosis of the data ECC decoder in instruction fetch unit is possible by fetching instruction from global RAM.

This operation can be used for the following ECC decoders:

- LSU Master data ECC: LSMxxxx of module BECCPE1 and BECCPE2
- IFU data ECC: IFxxxx of module BECCPE1 and BECCPE2

### (3) Self-diagnosis of the Address ECC check function

(a) Set the ECCTST bit in the address ECC test control register (see **Section 7.9.2.20**) to 1 to set test mode.

(b) Set the RWSEL bit in the address ECC test control register to select the DMA cycle to inject an error into address ECC bit.

(c) Set the ECCDAT[6:0] in the address ECC test data register (see **Section 7.9.2.21**) to an error data.

(d) For a channel of DMA,
In case of RWSEL= 0, Set up an address of area covered by target ECC decoder in a source address register. The destination address can be arbitrarily defined.
In case of RWSEL= 1, Set up an address of area covered by target ECC decoder in the destination address register. The source address can be arbitrarily defined.

After that, self-diagnosis of the address ECC decoder is possible by setting software DMA transfer request flag of the channel.

This operation can be used for the following ECC decoders:

- LSU Slave address ECC: LSSxxxx of module BECCPE1 and BECCPE2 (local RAM)
- P-Bus address ECC: module BECCPBAm (a peripheral on each P-Bus group)
- global RAM address ECC: xxxx_GRAMC in module ECCGRC (global RAM)
- address ECC for the bus width conversion in SIC: xxxx_VCI2VPI of module BECCSIC (a peripheral)
- address ECC for the bus width conversion in Flash interface: module BECCFLI (Code Flash)

A word in ( ) indicates the area that should be appointed as the destination or source.

## 28.2.10 ECC Function

### 28.2.10.1 ECC code for RAM Modules except Instruction cache RAM (data)

#### (1) Generating an Error Correcting Code

The following formulas are used to generate 7-bit error correcting codes (ecc6 to ecc2, ecc1z and ecc0z) for 32-bit data (d32 to d0). The lowest 2-bit of Error Correcting Code are inverted. With this measure, all 0s and all 1s of data and error correcting code is not a legal combination.

- ecc6 = (d13 ^ d12 ^ d11 ^ d10 ^ d9 ^ d8 ^ d7 ^ d6 ^ d5 ^ d4 ^ d3 ^ d2 ^ d1 ^ d0)
- ecc5 = (d23 ^ d22 ^ d21 ^ d20 ^ d19 ^ d18 ^ d17 ^ d16 ^ d15 ^ d14 ^ d3 ^ d2 ^ d1 ^ d0)
- ecc4 = (d29 ^ d28 ^ d27 ^ d26 ^ d25 ^ d24 ^ d17 ^ d16 ^ d15 ^ d14 ^ d7 ^ d6 ^ d5 ^ d4)
- ecc3 = (d31 ^ d30 ^ d26 ^ d25 ^ d24 ^ d20 ^ d19 ^ d18 ^ d14 ^ d10 ^ d9 ^ d8 ^ d4 ^ d0)
- ecc2 = (d31 ^ d30 ^ d28 ^ d27 ^ d24 ^ d22 ^ d21 ^ d18 ^ d15 ^ d12 ^ d11 ^ d8 ^ d5 ^ d1)
- ecc1z = ! (d30 ^ d29 ^ d27 ^ d25 ^ d23 ^ d21 ^ d19 ^ d16 ^ d13 ^ d11 ^ d9 ^ d6 ^ d2 ^ d0)
- ecc0z = ! (d31 ^ d29 ^ d28 ^ d26 ^ d23 ^ d22 ^ d20 ^ d17 ^ d13 ^ d12 ^ d10 ^ d7 ^ d3 ^ d0)

Where ^ denotes an exclusive OR.

#### (2) Error Correction

According to the following equation, 7-bit syndrome (synd6 to synd0) is generated through 32-bit data (d31 to d0) and 7-bit error correcting code (ecc6 to ecc2, ecc1z and ecc0z) which are read from the RAM.

$$\begin{pmatrix} 1000000\ 00000000\ 00000000\ 00111111\ 11111111 \\ 0100000\ 00000000\ 11111111\ 11000000\ 00001111 \\ 0010000\ 00111111\ 00000011\ 11000000\ 11110000 \\ 0001000\ 11000111\ 00011100\ 01000111\ 00010001 \\ 0000100\ 11011001\ 01100100\ 10011001\ 00100010 \\ 0000010\ 01101010\ 10101001\ 00101010\ 01000101 \\ 0000001\ 10110100\ 11010010\ 00110100\ 10001001 \end{pmatrix} \times \begin{pmatrix} ecc6 \\ ecc5 \\ : \\ !ecc1z \\ !ecc0z \\ d31 \\ d30 \\ : \\ d1 \\ d0 \end{pmatrix} = \begin{pmatrix} synd6 \\ synd5 \\ synd4 \\ synd3 \\ synd2 \\ synd1 \\ synd0 \end{pmatrix}$$

**Figure 28.2 Matrix for ECC Decoding.**

When synd6 to synd0 are all0, error correction is not performed. When some of synd6 to synd0 are 1, the target bit for correction is identified according to and the error bit is corrected. When synd6 to synd0 are the value not to indicate in, 2-bit error is detected.

**Table 28.151 Correspondence between Correction Target and Syndrome**

| synd[6:0] | Error Bit Position | synd[6:0] | Error Bit Position |
|---|---|---|---|
| 1000000 | ECC data bit 6 | 0101100 | RAM data bit 18 |
| 0100000 | ECC data bit 5 | 0110001 | RAM data bit 17 |
| 0010000 | ECC data bit 4 | 0110010 | RAM data bit 16 |
| 0001000 | ECC data bit 3 | 0110100 | RAM data bit 15 |
| 0000100 | ECC data bit 2 | 0111000 | RAM data bit 14 |
| 0000010 | ECC data bit 1 | 1000011 | RAM data bit 13 |
| 0000001 | ECC data bit 0 | 1000101 | RAM data bit 12 |
| 0001101 | RAM data bit 31 | 1000110 | RAM data bit 11 |
| 0001110 | RAM data bit 30 | 1001001 | RAM data bit 10 |
| 0010011 | RAM data bit 29 | 1001010 | RAM data bit 9 |
| 0010101 | RAM data bit 28 | 1001100 | RAM data bit 8 |
| 0010110 | RAM data bit 27 | 1010001 | RAM data bit 7 |
| 0011001 | RAM data bit 26 | 1010010 | RAM data bit 6 |
| 0011010 | RAM data bit 25 | 1010100 | RAM data bit 5 |
| 0011100 | RAM data bit 24 | 1011000 | RAM data bit 4 |
| 0100011 | RAM data bit 23 | 1100001 | RAM data bit 3 |
| 0100101 | RAM data bit 22 | 1100010 | RAM data bit 2 |
| 0100110 | RAM data bit 21 | 1100100 | RAM data bit 1 |
| 0101001 | RAM data bit 20 | 1101011 | RAM data bit 0 |
| 0101010 | RAM data bit 19 | 0000000 | No error |

### 28.2.10.2 EDC Function for Instruction cache RAM (data)

#### (1) Generating an Error Detecting Code

The following formulas are used to generate 8-bit error detecting codes (ecc7 to ecc2, ecc1z and ecc0z) for 64-bit data (d63 to d0). The lowest 2-bit of Error Correcting Code are inverted. With this measure, all 0s and all 1s of data and error correcting code is not a legal combination.

- ecc7 = (d63 ^ d62 ^ d61 ^ d60 ^ d48 ^ d31 ^ d30 ^ d29 ^ d28 ^ d27 ^ d26 ^ d25 ^ d24 ^ d21 ^ d20 ^ d19 ^ d18 ^ d14 ^ d13 ^ d10 ^ d9 ^ d7 ^ d5 ^ d3 ^ d1 ^ d0)
- ecc6 = (d63 ^ d61 ^ d59 ^ d57 ^ d56 ^ d55 ^ d54 ^ d53 ^ d52 ^ d40 ^ d23 ^ d22 ^ d21 ^ d20 ^ d19 ^ d18 ^ d17 ^ d16 ^ d13 ^ d12 ^ d11 ^ d10 ^ d6 ^ d5 ^ d2 ^ d1)
- ecc5 = (d62 ^ d61 ^ d58 ^ d57 ^ d55 ^ d53 ^ d51 ^ d49 ^ d48 ^ d47 ^ d46 ^ d45 ^ d44 ^ d32 ^ d15 ^ d14 ^ d13 ^ d12 ^ d11 ^ d10 ^ d9 ^ d8 ^ d5 ^ d4 ^ d3 ^ d2)
- ecc4 = (d61 ^ d60 ^ d59 ^ d58 ^ d54 ^ d53 ^ d50 ^ d49 ^ d47 ^ d45 ^ d43 ^ d41 ^ d40 ^ d39 ^ d38 ^ d37 ^ d36 ^ d24 ^ d7 ^ d6 ^ d5 ^ d4 ^ d3 ^ d2 ^ d1 ^ d0)
- ecc3 = (d63 ^ d62 ^ d61 ^ d60 ^ d59 ^ d58 ^ d57 ^ d56 ^ d53 ^ d52 ^ d51 ^ d50 ^ d46 ^ d45 ^ d42 ^ d41 ^ d39 ^ d37 ^ d35 ^ d33 ^ d32 ^ d31 ^ d30 ^ d29 ^ d28 ^ d16)
- ecc2 = (d55 ^ d54 ^ d53 ^ d52 ^ d51 ^ d50 ^ d49 ^ d48 ^ d45 ^ d44 ^ d43 ^ d42 ^ d38 ^ d37 ^ d34 ^ d33 ^ d31 ^ d29 ^ d27 ^ d25 ^ d24 ^ d23 ^ d22 ^ d21 ^ d20 ^ d8)
- ecc1z = !(d47 ^ d46 ^ d45 ^ d44 ^ d43 ^ d42 ^ d41 ^ d40 ^ d37 ^ d36 ^ d35 ^ d34 ^ d30 ^ d29 ^ d26 ^ d25 ^ d23 ^ d21 ^ d19 ^ d17 ^ d16 ^ d15 ^ d14 ^ d13 ^ d12 ^ d0)
- ecc0z = !(d56 ^ d39 ^ d38 ^ d37 ^ d36 ^ d35 ^ d34 ^ d33 ^ d32 ^ d29 ^ d28 ^ d27 ^ d26 ^ d22 ^ d21 ^ d18 ^ d17 ^ d15 ^ d13 ^ d11 ^ d9 ^ d8 ^ d7 ^ d6 ^ d5 ^ d4)

  Where ^ denotes an exclusive OR.

#### (2) Error Detection

According to the following equation, 8-bit syndrome (synd7 to synd0) is generated through 64-bit data (d63 to d0) and 8-bit error detecting code (ecc7 to ecc2, ecc1z and ecc0z) which are read from the RAM.

$$
\begin{pmatrix}
10000000\ 11110000\ 00000001\ 00000000\ 0000000011111111\ 00111100\ 01100110\ 10101011 \\
01000000\ 10101011\ 11110000\ 00000001\ 00000000\ 0000000011111111\ 00111100\ 01100110 \\
00100000\ 01100110\ 10101011\ 11110000\ 00000001\ 00000000\ 0000000011111111\ 00111100 \\
00010000\ 00111100\ 01100110\ 10101011\ 11110000\ 00000001\ 00000000\ 0000000011111111 \\
00001000\ 11111111\ 00111100\ 01100110\ 10101011\ 11110000\ 00000001\ 00000000\ 00000000 \\
00000100\ 00000000\ 11111111\ 00111100\ 01100110\ 10101011\ 11110000\ 00000001\ 00000000 \\
00000010\ 00000000\ 0000000011111111\ 00111100\ 01100110\ 10101011\ 11110000\ 00000001 \\
00000001\ 0000000100000000\ 0000000011111111\ 00111100\ 01100110\ 10101011\ 11110000
\end{pmatrix}
\times
\begin{pmatrix}
ecc6 \\ ecc5 \\ : \\ !ecc1z \\ !ecc0z \\ d31 \\ d30 \\ : \\ d1 \\ d0
\end{pmatrix}
=
\begin{pmatrix}
synd7 \\ synd6 \\ synd5 \\ synd4 \\ synd3 \\ synd2 \\ synd1 \\ synd0
\end{pmatrix}
$$

**Figure 28.3 Matrix for EDC Decoding.**

When synd7 to synd0 are all0, error detection is not performed. When some of synd7 to synd0 are 1, the target bit for detection is identified according to **Table 28.152**. When synd7 to synd0 are the value not to indicate in **Table 28.152**, 2-bit error is detected.

Table 28.152 Syndrome Codes and the Respective Error Bit Position

| synd[7:0] | Error Bit Position | synd[7:0] | Error Bit Position | synd[7:0] | Error Bit Position |
|---|---|---|---|---|---|
| 10000000 | ECC data bit 7 | 00101010 | RAM data bit 46 | 11000111 | RAM data bit 21 |
| 01000000 | ECC data bit 6 | 00111110 | RAM data bit 45 | 11000100 | RAM data bit 20 |
| 00100000 | ECC data bit 5 | 00100110 | RAM data bit 44 | 11000010 | RAM data bit 19 |
| 00010000 | ECC data bit 4 | 00010110 | RAM data bit 43 | 11000001 | RAM data bit 18 |
| 00001000 | ECC data bit 3 | 00001110 | RAM data bit 42 | 01000011 | RAM data bit 17 |
| 00000100 | ECC data bit 2 | 00011010 | RAM data bit 41 | 01001010 | RAM data bit 16 |
| 00000010 | ECC data bit 1 | 01010010 | RAM data bit 40 | 00100011 | RAM data bit 15 |
| 00000001 | ECC data bit 0 | 00011001 | RAM data bit 39 | 10100010 | RAM data bit 14 |
| 11001000 | RAM data bit 63 | 00010101 | RAM data bit 38 | 11100011 | RAM data bit 13 |
| 10101000 | RAM data bit 62 | 00011111 | RAM data bit 37 | 01100010 | RAM data bit 12 |
| 11111000 | RAM data bit 61 | 00010011 | RAM data bit 36 | 01100001 | RAM data bit 11 |
| 10011000 | RAM data bit 60 | 00001011 | RAM data bit 35 | 11100000 | RAM data bit 10 |
| 01011000 | RAM data bit 59 | 00000111 | RAM data bit 34 | 10100001 | RAM data bit 9 |
| 00111000 | RAM data bit 58 | 00001101 | RAM data bit 33 | 00100101 | RAM data bit 8 |
| 01101000 | RAM data bit 57 | 00101001 | RAM data bit 32 | 10010001 | RAM data bit 7 |
| 01001001 | RAM data bit 56 | 10001100 | RAM data bit 31 | 01010001 | RAM data bit 6 |
| 01100100 | RAM data bit 55 | 10001010 | RAM data bit 30 | 11110001 | RAM data bit 5 |
| 01010100 | RAM data bit 54 | 10001111 | RAM data bit 29 | 00110001 | RAM data bit 4 |
| 01111100 | RAM data bit 53 | 10001001 | RAM data bit 28 | 10110000 | RAM data bit 3 |
| 01001100 | RAM data bit 52 | 10000101 | RAM data bit 27 | 01110000 | RAM data bit 2 |
| 00101100 | RAM data bit 51 | 10000011 | RAM data bit 26 | 11010000 | RAM data bit 1 |
| 00011100 | RAM data bit 50 | 10000110 | RAM data bit 25 | 10010010 | RAM data bit 0 |
| 00110100 | RAM data bit 49 | 10010100 | RAM data bit 24 | 00000000 | No ECC error |
| 10100100 | RAM data bit 48 | 01000110 | RAM data bit 23 | | |
| 00110010 | RAM data bit 47 | 01000101 | RAM data bit 22 | | |

### 28.2.10.3 ECC Function for Code Flash

#### (1) Generating an Error Detecting Code

The following formulas are used to generate 9-bit error detecting codes (ecc8z, ecc7, ecc6z, ecc5, ecc4, ecc3z, ecc2, ecc1z and ecc0z) for 128-bit data (d127 to d0). 5-bits (ecc8, ecc6, ecc3, ecc1 and ecc0) of Error Correcting Code are inverted. With this measure, all 0s and all 1s of data and error correcting code is not a legal combination.

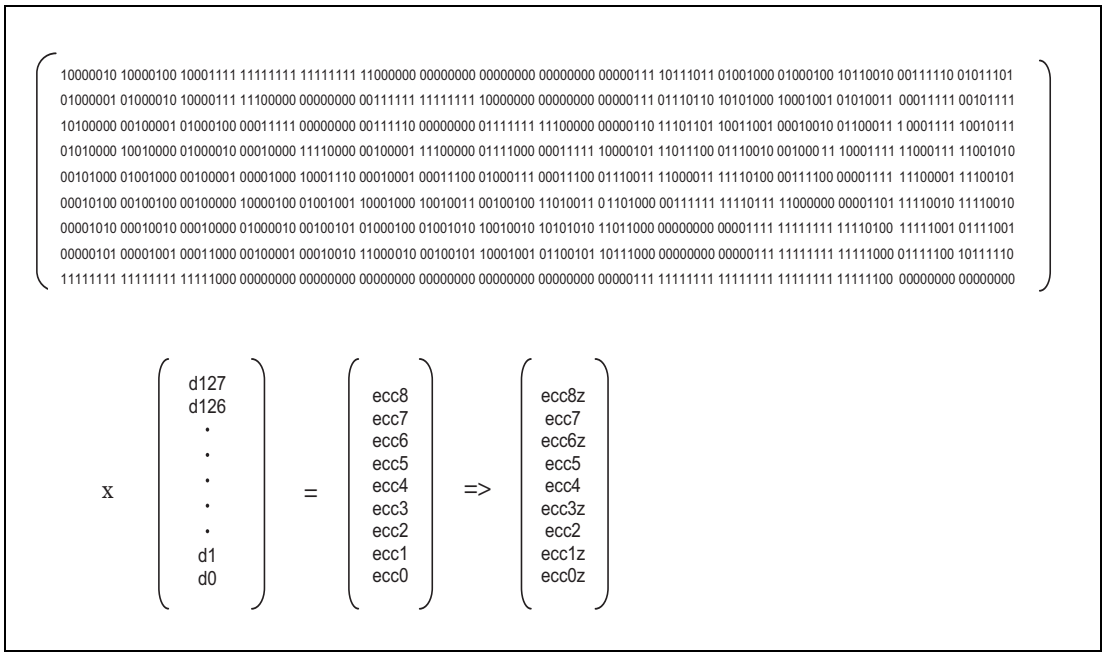



**Figure 28.4 Matrix for ECC Encoding.**

#### (2) Error Detection

According to the following equation, 9-bit syndrome (synd8 to synd0) is generated through 128-bit data (d127 to d0) and 9-bit error detecting code (ecc8z, ecc7, ecc6z, ecc5, ecc4, ecc3z, ecc2, ecc1z and ecc0z) which are read from the Code Flash.

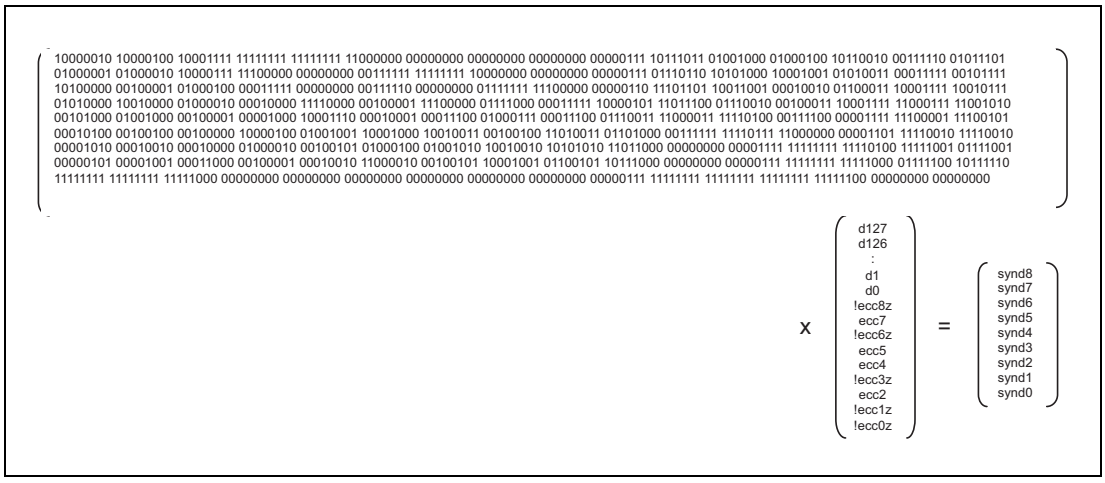



**Figure 28.5 Matrix for ECC Decoding**

When synd8 to synd0 are all0, error correction is not performed. When some of synd8 to synd0 are 1, the target bit for correction is identified according to **Table 28.153** and the error bit is corrected. When synd8 to synd0 are the value not to indicate in **Table 28.153**, 2-bit error is detected.

**Table 28.153 Correspondence between Correction Target and Syndrome (1/2)**

| synd[8:0] | Error Bit Position | synd[8:0] | Error Bit Position | synd[8:0] | Error Bit Position |
|---|---|---|---|---|---|
| 100000000 | ECC data bit 8 | 100010100 | Flash data bit 90 | 110101001 | Flash data bit 44 |
| 010000000 | ECC data bit 7 | 100010010 | Flash data bit 89 | 101101001 | Flash data bit 43 |
| 001000000 | ECC data bit 6 | 100001100 | Flash data bit 88 | 011101001 | Flash data bit 42 |
| 000100000 | ECC data bit 5 | 100001010 | Flash data bit 87 | 110011001 | Flash data bit 41 |
| 000010000 | ECC data bit 4 | 100000110 | Flash data bit 86 | 101011001 | Flash data bit 40 |
| 000001000 | ECC data bit 3 | 011100000 | Flash data bit 85 | 011011001 | Flash data bit 39 |
| 000000100 | ECC data bit 2 | 011010000 | Flash data bit 84 | 100111001 | Flash data bit 38 |
| 000000010 | ECC data bit 1 | 011001000 | Flash data bit 83 | 010111001 | Flash data bit 37 |
| 000000001 | ECC data bit 0 | 011000100 | Flash data bit 82 | 001111001 | Flash data bit 36 |
| 101000001 | Flash data bit 127 | 011000010 | Flash data bit 81 | 111000101 | Flash data bit 35 |
| 010100001 | Flash data bit 126 | 010110000 | Flash data bit 80 | 000011111 | Flash data bit 34 |
| 001010001 | Flash data bit 125 | 010101000 | Flash data bit 79 | 000101111 | Flash data bit 33 |
| 000101001 | Flash data bit 124 | 010100100 | Flash data bit 78 | 001001111 | Flash data bit 32 |
| 000010101 | Flash data bit 123 | 010100010 | Flash data bit 77 | 010001111 | Flash data bit 31 |
| 000001011 | Flash data bit 122 | 010011000 | Flash data bit 76 | 100001111 | Flash data bit 30 |
| 100000101 | Flash data bit 121 | 010010100 | Flash data bit 75 | 000110111 | Flash data bit 29 |
| 010000011 | Flash data bit 120 | 010010010 | Flash data bit 74 | 001010111 | Flash data bit 28 |
| 100100001 | Flash data bit 119 | 010001100 | Flash data bit 73 | 010010111 | Flash data bit 27 |
| 010010001 | Flash data bit 118 | 010001010 | Flash data bit 72 | 100010111 | Flash data bit 26 |
| 001001001 | Flash data bit 117 | 010000110 | Flash data bit 71 | 001100111 | Flash data bit 25 |
| 000100101 | Flash data bit 116 | 001110000 | Flash data bit 70 | 010100111 | Flash data bit 24 |
| 000010011 | Flash data bit 115 | 001101000 | Flash data bit 69 | 100100111 | Flash data bit 23 |
| 100001001 | Flash data bit 114 | 001100100 | Flash data bit 68 | 011000111 | Flash data bit 22 |
| 010000101 | Flash data bit 113 | 001100010 | Flash data bit 67 | 101000111 | Flash data bit 21 |
| 001000011 | Flash data bit 112 | 001011000 | Flash data bit 66 | 110000111 | Flash data bit 20 |
| 110000001 | Flash data bit 111 | 001010100 | Flash data bit 65 | 000111011 | Flash data bit 19 |
| 001100001 | Flash data bit 110 | 001010010 | Flash data bit 64 | 000111101 | Flash data bit 18 |
| 000011001 | Flash data bit 109 | 001001100 | Flash data bit 63 | 111110000 | Flash data bit 17 |
| 000000111 | Flash data bit 108 | 001001010 | Flash data bit 62 | 011111000 | Flash data bit 16 |
| 100000011 | Flash data bit 107 | 001000110 | Flash data bit 61 | 001111100 | Flash data bit 15 |
| 111000000 | Flash data bit 106 | 000111000 | Flash data bit 60 | 000111110 | Flash data bit 14 |
| 110100000 | Flash data bit 105 | 000110100 | Flash data bit 59 | 100011110 | Flash data bit 13 |
| 110010000 | Flash data bit 104 | 000110010 | Flash data bit 58 | 110001110 | Flash data bit 12 |
| 110001000 | Flash data bit 103 | 000101100 | Flash data bit 57 | 111000110 | Flash data bit 11 |
| 110000100 | Flash data bit 102 | 000101010 | Flash data bit 56 | 111100010 | Flash data bit 10 |
| 110000010 | Flash data bit 101 | 000100110 | Flash data bit 55 | 111101000 | Flash data bit 9 |
| 101100000 | Flash data bit 100 | 000011100 | Flash data bit 54 | 011110100 | Flash data bit 8 |

Table 28.153 Correspondence between Correction Target and Syndrome (2/2)

| synd[8:0] | Error Bit Position | synd[8:0] | Error Bit Position | synd[8:0] | Error Bit Position |
|---|---|---|---|---|---|
| 101010000 | Flash data bit 99 | 000011010 | Flash data bit 53 | 001111010 | Flash data bit 7 |
| 101001000 | Flash data bit 98 | 000010110 | Flash data bit 52 | 100111100 | Flash data bit 6 |
| 101000100 | Flash data bit 97 | 000001110 | Flash data bit 51 | 010011110 | Flash data bit 5 |
| 101000010 | Flash data bit 96 | 111100001 | Flash data bit 50 | 101001110 | Flash data bit 4 |
| 100110000 | Flash data bit 95 | 111010001 | Flash data bit 49 | 110100110 | Flash data bit 3 |
| 100101000 | Flash data bit 94 | 110110001 | Flash data bit 48 | 111010010 | Flash data bit 2 |
| 100100100 | Flash data bit 93 | 101110001 | Flash data bit 47 | 011101010 | Flash data bit 1 |
| 100100010 | Flash data bit 92 | 011110001 | Flash data bit 46 | 111010100 | Flash data bit 0 |
| 100011000 | Flash data bit 91 | 111001001 | Flash data bit 45 | 000000000 | No error |

### 28.2.10.4 ECC Function for Data Flash

#### (1) Generating an Error Detecting Code

The following formulas are used to generate 7-bit error detecting codes (ecc6z to ecc0z) for 32-bit data (d31 to d0). All bits of Error Correcting Code are inverted. With this measure, all 0s of data and error correcting code is not a legal combination.

- ecc6z = !(d13 ^ d12 ^ d11 ^ d10 ^ d9 ^ d8 ^ d7 ^ d6 ^ d5 ^ d4 ^ d3 ^ d2 ^ d1 ^ d0)
- ecc5z = !(d23 ^ d22 ^ d21 ^ d20 ^ d19 ^ d18 ^ d17 ^ d16 ^ d15 ^ d14 ^ d3 ^ d2 ^ d1 ^ d0)
- ecc4z = !(d29 ^ d28 ^ d27 ^ d26 ^ d25 ^ d24 ^ d17 ^ d16 ^ d15 ^ d14 ^ d7 ^ d6 ^ d5 ^ d4)
- ecc3z = !(d31 ^ d30 ^ d26 ^ d25 ^ d24 ^ d20 ^ d19 ^ d18 ^ d14 ^ d10 ^ d9 ^ d8 ^ d4 ^ d0)
- ecc2z = !(d31 ^ d30 ^ d28 ^ d27 ^ d24 ^ d22 ^ d21 ^ d18 ^ d15 ^ d12 ^ d11 ^ d8 ^ d5 ^ d1)
- ecc1z = !(d30 ^ d29 ^ d27 ^ d25 ^ d23 ^ d21 ^ d19 ^ d16 ^ d13 ^ d11 ^ d9 ^ d6 ^ d2 ^ d0)
- ecc0z = !(d31 ^ d29 ^ d28 ^ d26 ^ d23 ^ d22 ^ d20 ^ d17 ^ d13 ^ d12 ^ d10 ^ d7 ^ d3 ^ d0)

  Where ^ denotes an exclusive OR.

#### (2) Error Detection

According to the following equation, 7-bit syndrome (synd6 to synd0) is generated through 32-bit data (d31 to d0) and 7-bit error detecting code (ecc6z to ecc0z) which are read from the Data Flash.

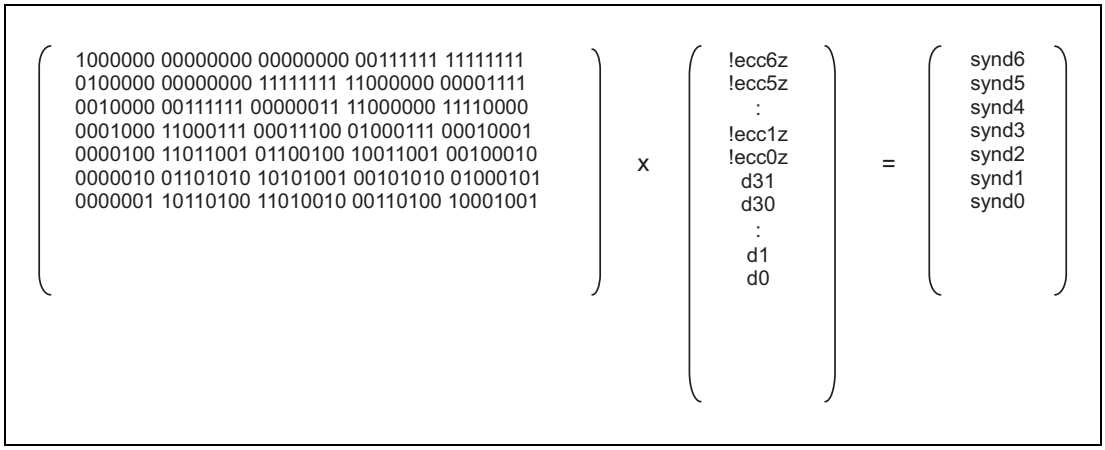



**Figure 28.6 Matrix for ECC Decoding**

When synd6 to synd0 are all0, error correction is not performed. When some of synd6 to synd0 are 1, the target bit for correction is identified according to **Table 28.154** and the error bit is corrected. When synd6 to synd0 are the value not to indicate in **Table 28.154**, 2-bit error is detected.

Table 28.154 Correspondence between Correction Target and Syndrome

| synd[6:0] | Error Bit Position | synd[6:0] | Error Bit Position |
|---|---|---|---|
| 1000000 | ECC data bit 6 | 0101100 | FLASH data bit 18 |
| 0100000 | ECC data bit 5 | 0110001 | FLASH data bit 17 |
| 0010000 | ECC data bit 4 | 0110010 | FLASH data bit 16 |
| 0001000 | ECC data bit 3 | 0110100 | FLASH data bit 15 |
| 0000100 | ECC data bit 2 | 0111000 | FLASH data bit 14 |
| 0000010 | ECC data bit 1 | 1000011 | FLASH data bit 13 |
| 0000001 | ECC data bit 0 | 1000101 | FLASH data bit 12 |
| 0001101 | FLASH data bit 31 | 1000110 | FLASH data bit 11 |
| 0001110 | FLASH data bit 30 | 1001001 | FLASH data bit 10 |
| 0010011 | FLASH data bit 29 | 1001010 | FLASH data bit 9 |
| 0010101 | FLASH data bit 28 | 1001100 | FLASH data bit 8 |
| 0010110 | FLASH data bit 27 | 1010001 | FLASH data bit 7 |
| 0011001 | FLASH data bit 26 | 1010010 | FLASH data bit 6 |
| 0011010 | FLASH data bit 25 | 1010100 | FLASH data bit 5 |
| 0011100 | FLASH data bit 24 | 1011000 | FLASH data bit 4 |
| 0100011 | FLASH data bit 23 | 1100001 | FLASH data bit 3 |
| 0100101 | FLASH data bit 22 | 1100010 | FLASH data bit 2 |
| 0100110 | FLASH data bit 21 | 1100100 | FLASH data bit 1 |
| 0101001 | FLASH data bit 20 | 1101011 | FLASH data bit 0 |
| 0101010 | FLASH data bit 19 | 0000000 | No error |

## 28.3 Lockstep

This product incorporates the CPU1/2 and DMA with the lockstep function to quickly detect failures without software interaction. The CPU1/2 and DMA executes the program using two different cores, that is, master core and checker core, and constantly compares the execution results of the two cores. When the results do not match, an error notification to the ECM takes place.

The lockstep function of the CPU1/2 and DMA features failure insertion, with which errors can be intentionally caused and thus self-diagnosis of the lockstep operation is possible.

### 28.3.1 List of Registers

**Table 28.155 List of Registers**

| Address | Symbol | Register Name | R/W | Value after reset | Access Size |
|---|---|---|---|---|---|
| FFFE ED00 | CMPTST0 | Comparator test register 0 | R/W | 0000 $0000_H$ | 8/16/32 |
| FFFE ED04 | CMPTST1 | Comparator test register 1 | R/W | 0000 $0000_H$ | 8/16/32 |
| FFC4 CA00 | PDMA_COMP_CNTRL | PDMA Comparator Error Injection Control Register | R/W | 0000 $0000_H$ | 32 |

CMPTST0 and CMPTST1 are placed in the CPU Peripheral of the CPU1/2. These registers can only be accessed by the CPU1/2.

## 28.3.2 Details of Registers

### 28.3.2.1 CMPTST0 — Comparator test register 0

CMPTST0 is test register 0 used for the lockstep function of the CPU1/2.

Combining CMPTST0 with CMPTST1 enables self-diagnosis of the lockstep function. The following gives an example of self-diagnosis procedure.

(1) Write arbitrary value to CMPTST0.

(2) Write a different value to CMPTST1.

(3) Read CMPTST0. The different values are read and sent to the master core and checker core.

(4) Using the values read, run the comparator to be diagnosed.

**Access:** This register can be Read/written in 32/16/8-bit units.

**Address:** FFFE ED00$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPTST0[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPTST0[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.156 CMPTST0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CMPTST0[31:0] | Write:<br>Data is written to each byte.<br>Read:<br>PE1/PE2: CMPTST0[31:0] value is read.<br>PE1C/PE2C: CMPTST1[31:0] value is read. |

### 28.3.2.2 CMPTST1 — Comparator test register 1

CMPTST1 is test register 1 used for the lockstep function of the CPU1/2.

Combining CMPTST1 with CMPTST0 enables self-diagnosis of the lockstep function.

**Access:** This register can be Read/written in 32/16/8-bit units.

**Address:** FFFE ED04$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPTST1[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPTST1[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.157 CMPTST1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CMPTST1[31:0] | Write:<br>Data is written to each byte.<br>Read:<br>PE1/PE2: CMPTST1[31:0] value is read.<br>PE1C/PE2C: CMPTST0[31:0] value is read. |

### 28.3.2.3 PDMA_COMP_CNTRL — PDMA Comparator Error Injection Control Register

This register (PDMA_COMP_CNTRL) can control the output signals on the checker side of the DMA.

A DMA comparison error can be generated by setting this register.

**Access:** This register can be read/written in 32-bit units.

**Address:** FFC4 CA00$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROT1 | PROT0 | DMACMPERR[29:16] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMACMPERR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.158 PDMA_COMP_CNTRL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | PROT[1:0] | Protection Bit<br>10: write is enable<br>Other: write is disable<br>These bits are always read as 0. |
| 29 to 0 | DMACMPERR [29:0] | A DMA comparison error can be generated by writing 111111_11111111_01111111_01111111 to DMACMPERR together with the PROT bit.<br>Clear all these bits to 0 if there is no need to generate this error. |

### 28.3.3 Usage Notes

Reading a register with a value that is undefined after a reset without initializing the register may lead to a CPU comparison error. Accordingly, we recommend initializing such registers with the desired settings.

**CAUTION**

**Be sure to use the startup routine to initialize these registers before referring to them since they are implicitly used by the C compiler.**

**refer to Section 3.4.6 register Initialization.**

# 28.4 Memory Protection

## 28.4.1 Overview

The overall memory protection architecture is shown in **Figure 28.7**. Each programmable core (bus master) has a Memory Protection Unit (MPU) that defines the access protection against the software. In addition, each resource (bus slave) has a guard that control the access by any bus master, including ones that do not have a MPU such as the DMA or the Flexray.

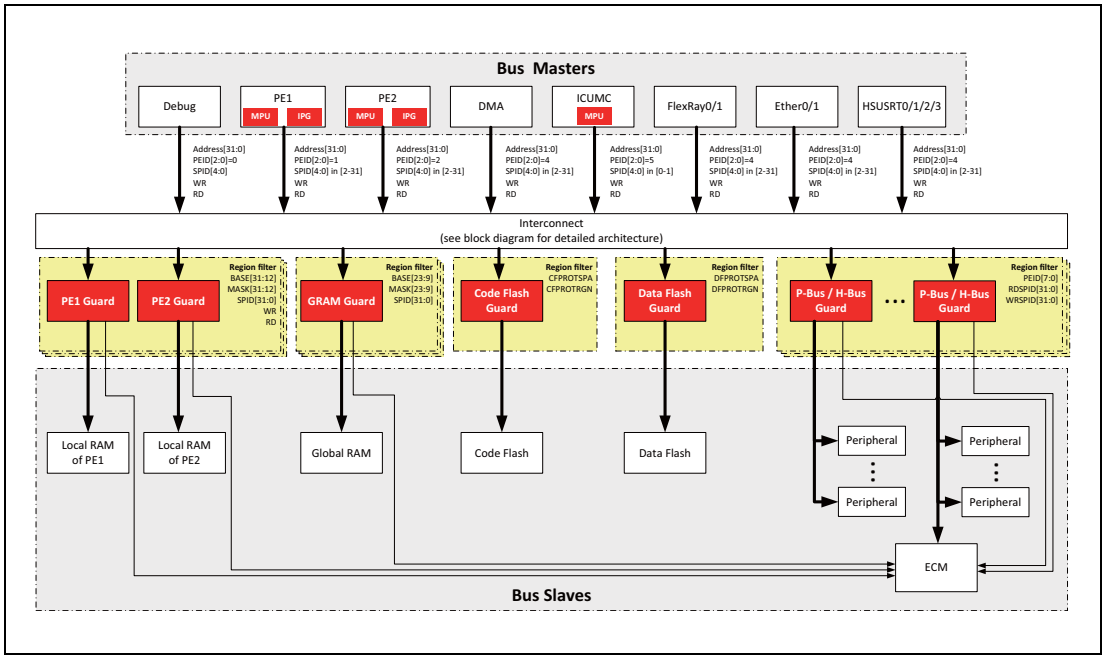



**Figure 28.7 Architecture of the memory protection.**

For this purpose, each bus access contains the following context information that identifies the bus master.

- Address: Memory location to access
- PEID: Processor Element ID (PEID)
- SPID: System Protection ID of the bus master
- WR: Access is a write request if set to 1
- RD: Access is a read request if set to 1

**Table 28.159 Bus Master Identifier**

| Identifier | Function |
|---|---|
| SPID | When the CPU makes an access, the system protection identifier SPID that is assigned to the CPU is indicated.<br>When the DMAC or DTS makes an access, the value of this identifier is the value in the channel master setting register.<br>When the other master makes an access, the value of this identifier is defined by each SPID register.<br>As 0 and 1 are reserved for the ICUMC, others except the ICUMC are limited to 2 to 31. |
| PEID | The access source bus master is indicated.<br>$000_B$: Reserved<br>$001_B$: PE1<br>$010_B$: PE2<br>$011_B$: Reserved<br>$100_B$: DMAC, DTS, Ethernet, FlexRay, HS-USRT<br>$101_B$: ICUMC<br>$110_B$: Reserved<br>$111_B$: Reserved |

Usage of the SPID is restricted depending if the bus master is secure or insecure. The security core ICUMC can only use the values 0 or 1. All other insecure cores can only use a value between 2 and 31. Therefore, it is not possible that PE1/2 identifies itself as a secure core, or that ICUMC identifies itself as a safe core. If a bus master is using a value that is not within its allowed range, the bus system overrides it by a value within its valid range. Please note that the register used to configure the SPID is not reflecting the overwritten value. It will still contain the original value. For the PE1, PE2 and ICUMC cores, the SPID is configured in the MCFG0 machine configuration register. It is initialized to 1 for ICUMC and to 2 for PE1/2 after reset.

For the peripherals that can act as bus masters (i.e., flexray, ethernet and high speed USART), there exists a register where the SPID can be configured and locked until next release. This registers will also not indicate the changed SPID value in case the bus system overrides it due to a violation of the allowed range. This registers can be read by the ICUMC or PE1/2, to check if the chosen SPID is correct.

The debug sub system can use any SPID. However, it cannot interfere with any function during normal operation, as it requires a secure handshake for activation.

Finally, the SPID that the DMA core uses can also be configured for each DMA/DTS channel by changing the channel master setting registers DMnnCM/DTSnnnCM. For all DMnnCM/DTSnnnCM registers, there exists one dedicated guard region. It can be used to disallow write access to all channel master setting registers by any bus master after the initial configuration has been performed. Similar to the peripheral bus masters, the chosen SPID can be controlled if the guard region allows read access by the given bus master.

Guards control the access to the resources based on the described bus access context information. Each guard supports a number of regions. Each region defines a filter which bus context is allowed to access.

There exist different types of guards depending on the memory resource: local RAM, global RAM, peripheral registers and flash. Each of those memory resources is protected by a guard. Hence, no bus master can access a resource without proper configuration of the guard. All guards except the flash guards support access protection based on SPID. Hence, read and write accesses to peripherals whose registers are located in P-Bus area and the H-Bus area can be protected. All registers located in the Local Peripheral Bus (LPB) area cannot be accessed by an external bus master.

The configuration registers of the guards are connected to the P-Bus. Access to this registers is also protected by a P-Bus guard. In order to disallow reconfiguration of any guard in user mode, a dedicated

SPID can be reserved that is only used by the cores in supervisor mode. The SPIDs of the peripherals and the DMA can be locked in hardware and checked such that they do not use the supervisor SPID. PE1/2 and ICUMC can only change their SPID in supervisor mode.

If a guard detects an illegal access, it reports it to the ECM. Additionally, information about the access context is stored in dedicated registers. The bus cycle is completed with an error response.

In case of an illegal write access, the data is not written. In case of an illegal read access, undefined data is returned.

#### 28.4.1.1 Protection mechanisms inside the processors

The PE has three mechanisms to protect its resources. The first mechanism is the Memory Protection Unit (MPU). It protects memory regions against illegal accesses by software. The second mechanism is the Internal Peripheral Guard (IPG). It protects registers of internal peripherals against accesses by software or external bus masters. The last mechanism is the PE Guard. It protects the whole PE against accesses by external bus masters.

**(1) Memory Protection Unit (MPU)**

Each CPU core has a dedicated Memory Protection Unit (MPU) to prevent unauthorized accesses to instruction and data. It is shown in red boxes with the label “MPU” in **Figure 28.7**. The MPU separates memory spaces, so that operation errors are constrained to the configured address space. Any unpermitted access to a protected memory area will raise a memory protection exception. As the MPU is integrated into the CPU core, it is aware of its pipeline and so it can generate precise exceptions on access violation.

If memory protection is enabled, all accesses which are not specifically enabled by an MPU area are prevented. The number of MPU areas is device dependent and generally between four and sixteen. Each MPU area is configured by writing dedicated system registers. Configuration of the MPU can only be performed in supervisor mode. Each CPU can only access and configure its own MPU.

Reconfiguration of MPU areas is typically done by the scheduler during context switch. One area may be setup to watch for stack overflows and underflows. Some MPU settings will usually apply globally to all threads and therefore these settings are not changed by the scheduler.

The CPU can operate in two privilege modes, user mode or supervisor mode. Privileges in user mode are restricted, while the supervisor mode permits all operations and accesses. There may be restrictions even in supervisor mode, but the CPU has the privilege to remove them if required. User mode does not permit to remove any restrictions.

The MPU provides up to 16 individual unified protection areas. The start and end address of each area can be specified with a granularity of one word (32-bit) and the size of each area may cover the whole address space. The access permissions for each area can be set to permit or prevent read accesses, write accesses and/or execute accesses. These access privileges can be specified independently for user mode and for supervisor mode. Areas may overlap, in which case permissions take precedence, i.e. if one of the areas permits the access, it is granted.

If the MPU detects an access which is not permitted, the access is not executed and a precise exception is raised. The exception handler can then decide to stop the violating thread or remove the restriction and resume execution.

The MPU only checks the addresses of the instruction and involved data for the currently executed instruction. Therefore, speculative access to instructions or data (e.g., during a cache prefetch) can only

trigger an illegal access if the instruction or data is used during instruction execution. All addresses that are not used by the CPU pipeline are also not checked by the MPU.

The peripheral address space is also part of the space, which is supervised by the MPU. Therefore the MPU must protect or unprotect a specific peripheral address space as needed. However, the MPU supervises only memory accesses of its own CPU core, not the accesses of other bus masters.

For a detailed functional description of the MPU, see **Section 3, CPU System** for PE1 and PE2, and Section X for ICUMC.

### (2) Internal Peripheral Guard (IPG)

Each PE has an internal peripheral guard (IPG) that protects the registers of peripherals inside the PE against invalid accesses. It is shown in red boxes with the label “IPG” in **Figure 28.7**. The protection granularity is based on a fixed number of peripheral groups. The assignment of peripherals to a group is also fixed.

The configuration is done via six registers IPGENUM and IPGPMTUM0 to 4.

If the guard detects an illegal access, it does not forward it to the peripheral. But the access context is stored in the two registers IPGECRUM and IPGADRUM. Additionally, a SYSERR exception is generated with the exception source code 0x18.

The IPGECRUM and IPGADRUM registers can be reset by writing a 0 to them.

For a detailed functional description of the IPG and the detailed register description, see **Section 3.2.4.2**.

### (3) PE Guard

Each PE has a guard that controls the access to the local RAM by other bus masters. It is shown as a red box with the label “PE Guard” in **Figure 28.7**. The access by bus masters can be controlled via their bus context for eight regions. Bus masters can only access the local RAM area if permission is granted. This includes the ICUMC (PEID = 5). So if ICUMC needs to read the local RAM, read allowance for SPID 0 or 1 must be configured. There is no hard-coded PEID filter for PEID = 5 available anymore. The address range and allowed bus masters of the region n are defined by the three registers PEGGnBA, PEGGnSPID and PEGGnMK. The PEGGnBA register defines the base address of the region n. The PEGGnMK register defines which bits of PEGGnBA are compared with the access address. If bit MASKm is cleared, bit BASEm is compared with bit m of the access address. Otherwise, the bit is ignored during the access check. Expressed in C notation, the address lies within the region if (address&(~MASK)&0xFFFFF000)) == (BASE&(~MASK)&0xFFFFF000)). Please note that the lower bit 11 to 0 of BASE and MASK are always zero.

Finally, the register PEGGnSPID defines the SPIDs that are allowed to access.

Each set bit k of PEGGnSPID register allows the access to the region n by the SPID k. Setting more than one bit allows to enable more than one SPID value at a time. For example, setting SPID to 0b0101 allows access with SPID = 0 and SPID = 2. The bits WR and RD of PEGGnBA control if write access or read access are allowed for region n. Finally, the bit EN decides if the region is considered during the access.

In case regions have overlapping address ranges, the resulting access permission is a union of the individual permissions. For example, if one region allows access for SPID 1 only, and the other one for SPID 2 only, the overlapping address ranges can be accessed by SPID 1 and 2.

The LOCK bit of PEGGnBA controls for each region n individually if the registers PEGGnBA, PEGGnMK and PEGGnSPID of region n can be changed. After reset, the registers can be re-written as long as LOCK remains zero. Once a one is written, all further write accesses are ignored.

Reconfiguration is only possible after the next reset. Access protection to the PE-Guard registers can be done via the LOCK bit, the MPU and the IPG.

After reset, region 0 is enabled and allows read only access to local RAM by SPID 0 and 1 (ICUMC).

If the guard detects an illegal access, it reports the violation to the Error Control Module (ECM) and reports the details about the access in three registers ERRSTATCTL, ERRSTAT, and ERRINFO. No exception is triggered. The register ERRSTAT indicates if an access violation has occurred. The register ERRINFO describes the context of the first access that triggered a violation. Finally, the register ERRSTATCTL is used to reset the ERRSTAT register. Please note that any further access violations are not stored as long as the ERR bit of ERRSTAT is set to one.

For a detailed functional description of the PE Guard, see **Section 3, CPU System** for PE1 and PE2. And please note that the ICUMC does not allow accesses by external bus masters. Hence, it also has no PE guard.

#### 28.4.1.2 Peripheral register protection with H-Bus Guard (HBG) and P-Bus Guard (PBG)

Each bus has slave guards that control which core can access the bus. The slave guards are shown in red boxes with the label H-Bus Guard (HBG) and P-Bus Guard (PBG) in **Figure 28.7**. Both types of slave guards use the same register interface. The region of protection is a peripheral instance, i.e., the configuration applies to all registers of the given peripheral instance. For each region n, there exists for each direction d (d = “RD” for reads or d = “WR” for writes) two registers FSGDxPRdPn and FSGDxPRdSn that configure the protection. The LOCK bit controls if these registers can be changed. After reset, the registers can be re-written as long as LOCK remains zero. Once a one is written, all further write accesses are ignored. Reconfiguration is only possible after the next reset.

The guard uses two filters in sequence to check if a core is allowed to access. The first filter checks the PEID. Each set bit k of PEID[k] allows access to the peripheral by the PEID k. After this filter is passed, the guard checks if the SPID matches the SPID filter. Each set bit k of SPID[k] allows access by SPID k. It is possible to set more than one bit in all filters, to enable more than one SPID/PEID value at a time. For example, setting PEID to 0b0110 allows access with PEID = 1 and PEID = 2.

If a guard detects an illegal access, it reports the violation to the Error Control Module (ECM) and reports the details about the access in three registers ERRSLVxCTL, ERRSLVxSTAT, and ERRSLVxTYPE. Depending on the bus topology, there exist multiple sets x of these three registers. The register ERRSLVxSTAT indicates if an access violation has occurred. The register ERRSLVxTYPE describes the context of the first access that triggered a violation. Finally, the register ERRSLVxCTL is used to reset the ERRSLVxSTAT register. Please note that any further access violations are not stored in ERRSLVxTYPE as long as the ERR bit of ERRSLVxSTAT is set to one.

For a detailed functional description of the peripheral register protection, see **Section 28.4.3** for PBG and **Section 28.4.4** for HBG.

#### 28.4.1.3 Global RAM protection

The global RAM is protected by a memory guard. It is shown as a red box with the label GRAM Guard in **Figure 28.7**. The guard can only protected against write accesses. Read accesses are always allowed by any bus master. The guard has eight memory regions that can be protected. The granularity of each region is 512 bytes. Each device has sufficient RAM to allow a fine granular control of the region size. For each region n, there exists four configuration registers MGDGRPROTn, MGDGRSPIDn, MGDGRBADn and MGDGRADVn. The register MGDGRPROTn configures the

contexts that are allowed to access. The LOCK bit controls if the registers of region n can be changed. After reset, the four registers can be re-written as long as LOCK remains zero. Once a one is written, all further write accesses are ignored. Reconfiguration is only possible after the next reset. The bit EN activates the protection for the given region n.

Each set bit k of MGDGRSPIDn allows the access to the peripheral by the SPID k. Setting more than one bit allows to enable more than one SPID value at a time. For example, setting SPID to 0b0101 allows access with SPID = 0 and SPID = 2. The UM bit defines for which modes the access is allowed.

The address range of the region n is defined by the two registers MGDGRBADn and MGDGRADVn. The MGDGRBADn register defines the base address of the region n. The MGDGRADVn register defines which bits of MGDGRBADn are compared with the access address. If bit ADV[m] is set, bit AD[m] is compared with bit m of the access address. Otherwise, the bit is ignored during the access check. Expressed in C notation, the address lies within the region if (address&ADV) == (AD&ADV).

Access to memory region n is granted if the access context passes the filter defined by this four registers. In case regions have overlapping address ranges, each region will be individually guarded. For example, if one region allows access for SPID 1 only, and the other one for SPID 2 only, the overlapping address regions cannot be accessed neither by SPID 1 nor 2.

After reset, no region is enabled. As long as no region is enabled, the GRAM guard allows read/write access by any bus master with any SPID.

If the guard detects an illegal access, it forwards it to the Error Control Module (ECM). The access context is stored in three registers MGDGRSCTL_, MGDGRSSTAT_, and MGDGRSTYPE_. The register MGDGRSSTAT indicates if an access violation has occurred. The register MGDGRSTYPE_ describes the context of the first access that triggered a violation. Finally, the register MGDGRSCTL_ is used to reset the MGDGRSSTAT_ register.

The registers of the memory guard are protected by a P-Bus guard.

For a detailed functional description of the GRAM Guard, see **Section 28.4.2**.

#### 28.4.1.4 Other Protection mechanisms

##### (1) External RAM protection

Protection of the external RAM is performed by a H-Bus guard. The external RAM supports four chip selects. For each chip select, there exists an own address range in the H-Bus area of the memory map. For each address range resp. chip select, one region is available in the H-Bus guard. The register interface of the H-Bus guard has already been described. For more information about the external memory controller, see **Section 15**.

##### (2) Flash memory protection

The code and data flash have their own guard. They are shown as red boxes with the label “Flash Guard” in **Figure 28.7**. From security point of view, these are not described in this section. Therefore see *Section for ICUMC* to get more information.

#### 28.4.1.5 Protectable memory regions

The number of supported memory regions and their granularity for each resource is summarized in **Table 28.160**.

**Table 28.160 Protection Region and Granularity**

| Protected resource | Protection mechanism | Number of memory regions | Minimum granularity of memory region [Byte] |
|---|---|---|---|
| Software access to address space of PE1 and PE2 | MPU | 16 | 4 |
| Access by other bus masters to local resources of PE1 and PE2 | PE Guard | 8 | 4096 |
| Peripherals on P-Bus and H-Bus | P-Bus / H-Bus guard | See **Section 4, Address Map** | Peripheral instance (chip select) |
| Global RAM | GRAM Guard | 8 | 512 |
| External RAM | H-Bus guard | 4 | 512 |
| Code and data flash | Flash Guard | 1 | N/A |

#### 28.4.1.6 Default MPU and guard configuration

The MPU of each PE and the ICUMC is disabled after reset. The H-Bus and P-Bus guard allow read and write access by PE1/2 only. Any SPID value is allowed to access. In case the P-Bus guard controls the access to the registers of another guard, it additionally allows access by ICUMC. All other guards require a configuration of the region first to define the protected address range. Therefore, their regions are disabled by default. See the reset values of the registers in the related sections for more information.

After reset, the ICUMC is the first core that starts software execution. It controls when the reset of the other cores is released. Hence, it can configure the guards as required before any other bus master can perform accesses.

Please note that all guards can be reconfigured during runtime, as long as configuration is not prevented by a lock bit or a slave guard. The configuration of the PE guard, the GRAM guard and all P-Bus guards can be changed while accesses are performed on the resource. The new configuration will not result in an undefined filter state. But the new filter configuration can get active at any time during an access to the protected resource. So it cannot be guaranteed if the current access is handled according to the old or the new filter configuration. However, the access itself is not corrupted or does result in a deadlock. Dynamic reconfiguration of the H-Bus guards (including the external memory interface) is not supported.This mean during reconfiguration accesses to the H-Bus peripherals from all bus masters is not allowed

## 28.4.2 GRG

This product is provided with 4-channel GRG, which is described in detail in the following sections.

### 28.4.2.1 List of Registers

Table 28.161 List of Registers (1/2)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFC4 $9000_H$ | MGDGRPROT0 | Global-RAM FS Guard Protection Setting Register0 | R/W | 07FF $FE10_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (1)** | |
| FFC4 $9004_H$ | MGDGRSPID0 | Global-RAM FS Guard SPID Setting Register0 | R/W | FFFF $FFFF_H$ | 32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (2)** | |
| FFC4 $9008_H$ | MGDGRBAD0 | Global-RAM FS Guard Base Address Register0 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (3)** | |
| FFC4 $900C_H$ | MGDGRADV0 | Global-RAM FS Guard Address Valid Register0 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (4)** | |
| FFC4 $9010_H$ | MGDGRPROT1 | Global-RAM FS Guard Protection Setting Register1 | R/W | 07FF $FE10_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (1)** | |
| FFC4 $9014_H$ | MGDGRSPID1 | Global-RAM FS Guard SPID Setting Register1 | R/W | FFFF $FFFF_H$ | 32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (2)** | |
| FFC4 $9018_H$ | MGDGRBAD1 | Global-RAM FS Guard Base Address Register1 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (3)** | |
| FFC4 $901C_H$ | MGDGRADV1 | Global-RAM FS Guard Address Valid Register1 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (4)** | |
| FFC4 $9020_H$ | MGDGRPROT2 | Global-RAM FS Guard Protection Setting Register2 | R/W | 07FF $FE10_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (1)** | |
| FFC4 $9024_H$ | MGDGRSPID2 | Global-RAM FS Guard SPID Setting Register2 | R/W | FFFF $FFFF_H$ | 32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (2)** | |
| FFC4 $9028_H$ | MGDGRBAD2 | Global-RAM FS Guard Base Address Register2 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (3)** | |
| FFC4 $902C_H$ | MGDGRADV2 | Global-RAM FS Guard Address Valid Register2 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (4)** | |
| FFC4 $9030_H$ | MGDGRPROT3 | Global-RAM FS Guard Protection Setting Register3 | R/W | 07FF $FE10_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (1)** | |
| FFC4 $9034_H$ | MGDGRSPID3 | Global-RAM FS Guard SPID Setting Register3 | R/W | FFFF $FFFF_H$ | 32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (2)** | |
| FFC4 $9038_H$ | MGDGRBAD3 | Global-RAM FS Guard Base Address Register3 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (3)** | |
| FFC4 $903C_H$ | MGDGRADV3 | Global-RAM FS Guard Address Valid Register3 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (4)** | |
| FFC4 $9040_H$ | MGDGRPROT4 | Global-RAM FS Guard Protection Setting Register4 | R/W | 07FF $FE10_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (1)** | |
| FFC4 $9044_H$ | MGDGRSPID4 | Global-RAM FS Guard SPID Setting Register4 | R/W | FFFF $FFFF_H$ | 32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (2)** | |
| FFC4 $9048_H$ | MGDGRBAD4 | Global-RAM FS Guard Base Address Register4 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (3)** | |
| FFC4 $904C_H$ | MGDGRADV4 | Global-RAM FS Guard Address Valid Register4 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (4)** | |
| FFC4 $9050_H$ | MGDGRPROT5 | Global-RAM FS Guard Protection Setting Register5 | R/W | 07FF $FE10_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (1)** | |
| FFC4 $9054_H$ | MGDGRSPID5 | Global-RAM FS Guard SPID Setting Register5 | R/W | FFFF $FFFF_H$ | 32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (2)** | |
| FFC4 $9058_H$ | MGDGRBAD5 | Global-RAM FS Guard Base Address Register5 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (3)** | |
| FFC4 $905C_H$ | MGDGRADV5 | Global-RAM FS Guard Address Valid Register5 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (4)** | |
| FFC4 $9060_H$ | MGDGRPROT6 | Global-RAM FS Guard Protection Setting Register6 | R/W | 07FF $FE10_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (1)** | |
| FFC4 $9064_H$ | MGDGRSPID6 | Global-RAM FS Guard SPID Setting Register6 | R/W | FFFF $FFFF_H$ | 32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (2)** | |
| FFC4 $9068_H$ | MGDGRBAD6 | Global-RAM FS Guard Base Address Register6 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (3)** | |
| FFC4 $906C_H$ | MGDGRADV6 | Global-RAM FS Guard Address Valid Register6 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (4)** | |

**Table 28.161 List of Registers (2/2)**

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFC4 $9070_H$ | MGDGRPROT7 | Global-RAM FS Guard Protection Setting Register7 | R/W | 07FF $FE10_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (1)** | |
| FFC4 $9074_H$ | MGDGRSPID7 | Global-RAM FS Guard SPID Setting Register7 | R/W | FFFF $FFFF_H$ | 32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (2)** | |
| FFC4 $9078_H$ | MGDGRBAD7 | Global-RAM FS Guard Base Address Register7 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (3)** | |
| FFC4 $907C_H$ | MGDGRADV7 | Global-RAM FS Guard Address Valid Register7 | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (4)** | |
| FFC4 $9100_H$ | MGDGRSCTL_VCI2GRAM | Global-RAM FS Guard Control Register (VCI2GRAM) | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (5)** | |
| FFC4 $9104_H$ | MGDGRSSTAT_VCI2GRAM | Global-RAM FS Guard Error Status Register (VCI2GRAM) | R | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (6)** | |
| FFC4 $910C_H$ | MGDGRSTYPE_VCI2GRAM | Global-RAM FS Guard Error Access Type Register (VCI2GRAM) | R | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (7)** | |
| FFC4 $9200_H$ | MGDGRSCTL_PE1 | Global-RAM FS Guard Control Register (PE1) | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP1 | **28.4.2.2 (5)** | |
| FFC4 $9204_H$ | MGDGRSSTAT_PE1 | Global-RAM FS Guard Error Status Register (PE1) | R | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP1 | **28.4.2.2 (6)** | |
| FFC4 $920C_H$ | MGDGRSTYPE_PE1 | Global-RAM FS Guard Error Access Type Register (PE1) | R | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP1 | **28.4.2.2 (7)** | |
| FFC4 $9300_H$ | MGDGRSCTL_PE2 | Global-RAM FS Guard Control Register (PE2) | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP2 | **28.4.2.2 (5)** | except P1M-C |
| FFC4 $9304_H$ | MGDGRSSTAT_PE2 | Global-RAM FS Guard Error Status Register (PE2) | R | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP2 | **28.4.2.2 (6)** | except P1M-C |
| FFC4 $930C_H$ | MGDGRSTYPE_PE2 | Global-RAM FS Guard Error Access Type Register (PE2) | R | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP2 | **28.4.2.2 (7)** | except P1M-C |
| FFC4 $9700_H$ | MGDGRSCTL_AXI2GRAM | Global-RAM FS Guard Control Register (AXI2GRAM) | R/W | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (5)** | |
| FFC4 $9704_H$ | MGDGRSSTAT_AXI2GRAM | Global-RAM FS Guard Error Status Register (AXI2GRAM) | R | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (6)** | |
| FFC4 $970C_H$ | MGDGRSTYPE_AXI2GRAM | Global-RAM FS Guard Error Access Type Register (AXI2GRAM) | R | 0000 $0000_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.2.2 (7)** | |

- MGDGRPROTn, MGDGRSPIDn, MGDGRBADn, and MGDGRADVn set the protection specifications for each channel (n: 0 to 7).
- MGDGRSCTL_*, MGDGRSSTAT_*, and MGDGRSTYPE_* indicate error information on each access port: “_VCI2GRAM” represents access from the system interconnect to the Global RAM, “_PE1”, and “_PE2” represents access from the CPU1 respectively CPU2 to the Global RAM, “_AXI2GRAM” represents access from the H-Bus to the Global RAM,

### 28.4.2.2 Details of Registers

#### (1) MGDGRPROT*n* — Global-RAM FS Guard Protection Setting Register *n* (*n* = 0 to 7)

**Access:** This register can be Read/written in 32/16/8-bit units.

**Address:** MGDGRPROT0: FFC4 $9000_H$
MGDGRPROT1: FFC4 $9010_H$
MGDGRPROT2: FFC4 $9020_H$
MGDGRPROT3: FFC4 $9030_H$
MGDGRPROT4: FFC4 $9040_H$
MGDGRPROT5: FFC4 $9050_H$
MGDGRPROT6: FFC4 $9060_H$
MGDGRPROT7: FFC4 $9070_H$

**Value after reset:** 07FF $FE10_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOCK | EN | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.162 MGDGRPROT*n* register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | LOCK | Lock Bit<br>0: Registers can be-written<br>1: Any further write to MGDGRPROTn and MGDGRSPIDn are ignored.<br>This bit can only be cleared by reset. |
| 30 | EN | Guard Enable<br>0: Guard Disable<br>1: Guard Enable |
| 29 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (2) MGDGRSPID*n* — Global-RAM FS Guard SPID Setting Register *n* (*n* = 0 to 7)

**Access:** This register can be Read/written in 32/16/8-bit units.

**Address:** MGDGRSPID0: FFC4 $9004_H$
MGDGRSPID1: FFC4 $9014_H$
MGDGRSPID2: FFC4 $9024_H$
MGDGRSPID3: FFC4 $9034_H$
MGDGRSPID4: FFC4 $9044_H$
MGDGRSPID5: FFC4 $9054_H$
MGDGRSPID6: FFC4 $9064_H$
MGDGRSPID7: FFC4 $9074_H$

**Value after reset:** FFFF $FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPID[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.163 MGDGRSPID*n* register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SPID[31:0] | In the case of SPID[x] = 0: Access with SPID = x is not allowed.<br>In the case of SPID[x] = 1: Access with SPID = x is allowed. |

### (3) MGDGRBAD*n* — Global-RAM FS Guard Base Address Register *n* (*n* = 0 to 7)

**Access:** This register can be Read/written in 32/16/8-bit units.

**Address:** MGDGRBAD0: FFC4 $9008_H$
MGDGRBAD1: FFC4 $9018_H$
MGDGRBAD2: FFC4 $9028_H$
MGDGRBAD3: FFC4 $9038_H$
MGDGRBAD4: FFC4 $9048_H$
MGDGRBAD5: FFC4 $9058_H$
MGDGRBAD6: FFC4 $9068_H$
MGDGRBAD7: FFC4 $9078_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | AD[20:16] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AD[15:9] | | | | | | | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R |

**Table 28.164 MGDGRBAD*n* register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 20 to 9 | AD[20:9] | Compare Base Address |
| 8 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (4) MGDGRADV*n* — Global-RAM FS Guard Address Valid Register *n* (*n* = 0 to 7)

**Access:** This register can be Read/written in 32/16/8-bit units.

**Address:** MGDGRADV0: FFC4 900C$_H$
MGDGRADV1: FFC4 901C$_H$
MGDGRADV2: FFC4 902C$_H$
MGDGRADV3: FFC4 903C$_H$
MGDGRADV4: FFC4 904C$_H$
MGDGRADV5: FFC4 905C$_H$
MGDGRADV6: FFC4 906C$_H$
MGDGRADV7: FFC4 907C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | ADV[20:16] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADV[15:9] | | | | | | | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R |

**Table 28.165 MGDGRADV*n* register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 20 to 9 | ADV[20:9] | Compare Address Valid<br>MGDGRADVn[x] = 0 indicates that the bit is not to be compared. So, all “0” of MGDGRADVn[x] means that the access is the scope of access protection in any address. |
| 8 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (5) MGDGRSCTL_VCI2GRAM/PE1/PE2/AXI2GRAM — Global-RAM FS Guard Control Register

**Access:** These registers can be Read/written in 32-bit units.

**Address:** MGDGRSCTL_VCI2GRAM: FFC4 $9100_H$
MGDGRSCTL_PE1: FFC4 $9200_H$
MGDGRSCTL_PE2: FFC4 $9300_H$
MGDGRSCTL_AXI2GRAM: FFC4 $9700_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLRO | CLRE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 28.166 MGDGRSCTL_VCI2GRAM/PE1/PE2/AXI2GRAM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | CLRO | Clear the OVF bit of MGDGRSSTAT_VCI2GRAM/PE1/PE2/AXI2GRAM by writing this bit to “1”.<br>Read value:<br>0: Clear is completed<br>1: Clear is on execution |
| 0 | CLRE | Clear the ERR bit of MGDGRSSTAT_ VCI2GRAM/PE1/PE2/AXI2GRAM by writing this bit to “1”.<br>Read value:<br>0: Clear is completed<br>1: Clear is on execution |

| CLRO | CLRE | Function |
|---|---|---|
| 0 | 0 | Not Clear both bit |
| 0 | 1 | Not Clear both bit (This setting is ignore) |
| 1 | 0 | OVF bit is Cleared |
| 1 | 1 | Both bit is Cleared |

### (6) MGDGRSSTAT_VCI2GRAM/PE1/PE2/AXI2GRAM — Global-RAM FS Guard Error Status Register

**Access:** These registers can be read only in 32-bit units.

**Address:** MGDGRSSTAT_VCI2GRAM: FFC4 $9104_H$
MGDGRSSTAT_PE1: FFC4 $9204_H$
MGDGRSSTAT_PE2: FFC4 $9304_H$
MGDGRSSTAT_AXI2GRAM: FFC4 $9704_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF | ERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.167 MGDGRSSTAT_VCI2GRAM/PE1/PE2/AXI2GRAM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1 | OVF | Error Overflow Status Flag<br>0: No Overflow<br>1: Error Overflow |
| 0 | ERR | Error Status Flag<br>0: No Error<br>1: Error |

### (7) MGDGRSTYPE_VCI2GRAM/PE1/PE2/AXI2GRAM — Global-RAM FS Guard Error Access Type Register

**Access:** These registers can be read only in 32/16/8-bit units.

**Address:** MGDGRSTYPE_VCI2GRAM: FFC4 910C$_H$
MGDGRSTYPE_PE1: FFC4 920C$_H$
MGDGRSTYPE_PE2: FFC4 930C$_H$
MGDGRSTYPE_AXI2GRAM: FFC4 970C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ERRCAUSE | — | — | — | SPID[4:0] | | | | | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID[2:0] | | | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.168 MGDGRSTYPE_VCI2GRAM/PE1/PE2/AXI2GRAM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | When read, the value after reset is read. |
| 28 | ERRCAUSE | This bit indicates if error cause was guard or illegal access<br>0: guard error<br>1: unmapped area access |
| 27 to 25 | Reserved | When read, the value after reset is read. |
| 24 to 20 | SPID[4:0] | SPID at Guard / unmapped area access error has occurred |
| 19, 18 | Reserved | When read, the value after reset is read. |
| 17, 16 | Reserved | These bits are undefined. |
| 15 to 13 | PEID[2:0] | PEID at Guard / unmapped area access error has occurred |
| 12 to 8 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 7, 6 | Reserved | These bits are undefined. |
| 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 to 0 | Reserved | These bits are undefined. |

### 28.4.3 PBG

The PBG module is divided into several PBG groups, each of which is provided with a maximum of 16 protection channels.When more than 16 modules belong to the same PBG group. The PBG group is divide into an additional PBG group, for example PBG1 #0 and PBG1 #1. Both PBG groups still share the same Error Status Register. A single PBG channel can designate the access against which a single peripheral circuit should be protected. Each PBG group can hold the information of the access that has been rejected.

The following table lists the peripheral circuits to be protected, the corresponding PBG group names, and the PBG channel numbers.

**Table 28.169 PBG Group (1/8)**

| PBG group | PBG channel number | Guard Group | purpose (R or W) | Module to be protected | Guard Register | Error status of PBG group | Note |
|---|---|---|---|---|---|---|---|
| APBGRD_PFSS1 | 0 | PBG | R | • P-Bus guard control (APBGRD_PFSS1, APBGRD_PFSS0) | APBFSGDPROT_PBG_A<br>APBFSGDSPID_PBG_A | ERRSLVCTL_PFSS1<br>ERRSLVSTAT_PFSS1<br>ERRSLVTYPE_PFSS1<br>(Included in SP4) | |
| | 1 | | W | | APBFSGDPROT_PBG_B<br>APBFSGDSPID_PBG_B | | |
| | 2 | SP1 | R | • Code Flash guard for PE1<br>• Global RAM guard for PE1<br>• ECC Controller for PE1<br>• PEG Error capture for PE1 | APBFSGDPROT_SP1_A<br>APBFSGDSPID_SP1_A | | |
| | 3 | | W | | APBFSGDPROT_SP1_B<br>APBFSGDSPID_SP1_B | | |
| | 4 | SP2 | R | • Code Flash guard for PE2<br>• Global RAM guard for PE2<br>• ECC Controller for PE2<br>• PEG Error capture for PE2 | APBFSGDPROT_SP2_A<br>APBFSGDSPID_SP2_A | | except P1M-C |
| | 5 | | W | | APBFSGDPROT_SP2_B<br>APBFSGDSPID_SP2_B | | except P1M-C |
| | 6 | SP4 | R | • Code flash guard for VCI2CFB<br>• Global RAM guard without suffix PE1 and PE2<br>• All ECC Controller outside of PEn<br>• Code Flash Interface<br>• PFSS System controller<br>• Error Slave P-bus Group<br>• Error Slave P-bus Area Error<br>• PDMA Comparator Control | APBFSGDPROT_SP4_A<br>APBFSGDSPID_SP4_A | | |
| | 7 | | W | | APBFSGDPROT_SP4_B<br>APBFSGDSPID_SP4_B | | |
| | 8 | reserved | | | | | |
| | 9 | reserved | | | | | |
| | 10 | reserved | | | | | |
| | 11 | reserved | | | | | |
| | 12 | reserved | | | | | |
| | 13 | reserved | | | | | |
| | 14 | reserved | | | | | |
| | 15 | reserved | | | | | |

Table 28.169 PBG Group (2/8)

| PBG group | PBG channel number | Guard Group | purpose (R or W) | Module to be protected | Guard Register | Error status of PBG group | Note |
|---|---|---|---|---|---|---|---|
| APBGRD_PFSS0 | 0 | PDMCM | R | • PDMA (DMnnCM and DTSnnCM) | APBFSGDPROT_PDMACM_A<br>APBFSGDSPID_PDMACM_A | ERRSLVCTL_PFSS0<br>ERRSLVSTAT_PFSS0<br>ERRSLVTYPE_PFSS0<br>(Included in SP4) | |
| | 1 | | W | | APBFSGDPROT_PDMACM_B<br>APBFSGDSPID_PDMACM_B | | |
| | 2 | PDMCH | R | • PDMA (Other) | APBFSGDPROT_PDMACH_A<br>APBFSGDSPID_PDMACH_A | | |
| | 3 | | W | | APBFSGDPROT_PDMACH_B<br>APBFSGDSPID_PDMACH_B | | |
| | 4 | INTC2 | R | • INTC2 | APBFSGDPROT_INTC2_A<br>APBFSGDSPID_INTC2_A | | |
| | 5 | | W | | APBFSGDPROT_INTC2_B<br>APBFSGDSPID_INTC2_B | | |
| | 6 | reserved | | | | | |
| | 7 | reserved | | | | | |
| | 8 | reserved | | | | | |
| | 9 | reserved | | | | | |
| | 10 | reserved | | | | | |
| | 11 | reserved | | | | | |
| | 12 | reserved | | | | | |
| | 13 | reserved | | | | | |
| | 14 | reserved | | | | | |
| | 15 | reserved | | | | | |

Table 28.169 PBG Group (3/8)

| PBG group | PBG channel number | Guard Group | purpose (R or W) | Module to be protected | Guard Register | Error status of PBG group | Note |
|---|---|---|---|---|---|---|---|
| PBG1 #0 | 0 | PG1-PBG1FIL0<br>PG1-PBG1FIL1<br>PG1-PBG1ERR0 | R | • P-Bus guard control and error register (P-Bus group 1; #0)<br>• P-Bus guard control and error register (P-Bus group 1; #1)<br>• ERRSLV for P-Bus guard 1; #0 & #1 | FSGD1APROT00<br>FSGD1ASPID00 | ERRSLV1CTL<br>ERRSLV1STAT<br>ERRSLV1TYPE<br>(Included in PG1-PBG1ERR0) | |
| | 1 | | W | | FSGD1APROT01<br>FSGD1ASPID01 | | |
| | 2 | PG1-Startup | R | • ECC on Bus (P-Bus group1)<br>• ECC Controller for Peripheral RAM (MCAN0, MCAN2,CSIH1/3)<br>• | FSGD1APROT02<br>FSGD1ASPID02 | | |
| | 3 | | W | | FSGD1APROT03<br>FSGD1ASPID03 | | |
| | 4 | PG1-RLIN3 | R | • RLIN31/RLIN33 | FSGD1APROT04<br>FSGD1ASPID04 | | |
| | 5 | | W | | FSGD1APROT05<br>FSGD1ASPID05 | | |
| | 6 | PG1-ECM1 | R | • ECM1 | FSGD1APROT06<br>FSGD1ASPID06 | | except P1M-C |
| | 7 | | W | | FSGD1APROT07<br>FSGD1ASPID07 | | except P1M-C |
| | 8 | reserved | | | | | |
| | 9 | reserved | | | | | |
| | 10 | reserved | | | | ERRSLV1CTL<br>ERRSLV1STAT<br>ERRSLV1TYPE<br>(Included in PG1-PBG1ERR0) | |
| | 11 | reserved | | | | | |
| | 12 | reserved | | | | | |
| | 13 | reserved | | | | | |
| | 14 | reserved | | | | | |
| | 15 | reserved | | | | | |

Table 28.169 PBG Group (4/8)

| PBG group | PBG channel number | Guard Group | purpose (R or W) | Module to be protected | Guard Register | Error status of PBG group | Note |
|---|---|---|---|---|---|---|---|
| PBG1 #1 | 0 | PG1-MCAN0 | R | • MCAN0 | FSGD1BPROT00<br>FSGD1BSPID00 | ERRSLV1CTL<br>ERRSLV1STAT<br>ERRSLV1TYPE<br>(Included in PG1-PBG1ERR0) | |
| | 1 | | W | | FSGD1BPROT01<br>FSGD1BSPID01 | | |
| | 2 | PG1-MCAN2 | R | • MCAN2 | FSGD1BPROT02<br>FSGD1BSPID02 | | except P1M-C, P1H-C (4MB) |
| | 3 | | W | | FSGD1BPROT03<br>FSGD1BSPID03 | | except P1M-C, P1H-C (4MB) |
| | 4 | reserved | | | | | |
| | 5 | reserved | | | | | |
| | 6 | PG1-HSUSRT1 | R | • HSUS1 | FSGD1BPROT06<br>FSGD1BSPID06 | | |
| | 7 | | W | | FSGD1BPROT07<br>FSGD1BSPID07 | | |
| | 8 | PG1-HSUSRT3 | R | • HSUS3 | FSGD1BPROT08<br>FSGD1BSPID08 | | except P1M-C, P1H-C (4MB, BGA-156) |
| | 9 | | W | | FSGD1BPROT09<br>FSGD1BSPID09 | | except P1M-C, P1H-C (4MB, BGA-156) |
| | 10 | PG1-CSIH1 | R | • CSIH1 | FSGD1BPROT10<br>FSGD1BSPID10 | | |
| | 11 | | W | | FSGD1BPROT11<br>FSGD1BSPID11 | | |
| | 12 | PG1-CSIH3 | R | • CSIH3 | FSGD1BPROT12<br>FSGD1BSPID12 | | |
| | 13 | | W | | FSGD1BPROT13<br>FSGD1BSPID13 | | |
| | 14 | reserved | | | | | |
| | 15 | reserved | | | | | |
| PBG2 | 0 | PG2-PBG2FIL<br>PG2-PBG2ERR0 | R | • P-Bus guard control and error register (P-Bus group 2; #0)<br>• ERRSLV for P-Bus guard 2; #0 | FSGD2APROT00<br>FSGD2ASPID00 | ERRSLV2CTL<br>ERRSLV2STAT<br>ERRSLV2TYPE<br>(Included in PG2-PBG2ERR0) | |
| | 1 | | W | | FSGD2APROT01<br>FSGD2ASPID01 | | |
| | 2 | PG2-Startup | R | • ECC on Bus (P-Bus group2)<br>• ECC Controller for Peripheral RAM (GTM) | FSGD2APROT02<br>FSGD2ASPID02 | | |
| | 3 | | W | | FSGD2APROT03<br>FSGD2ASPID03 | | |
| | 4 | PG2-GTM0 | R | • GTM | FSGD2APROT04<br>FSGD2ASPID04 | | |
| | 5 | | W | | FSGD2APROT05<br>FSGD2ASPID05 | | |
| | 6 | PG2-SSTM0 | R | • STM0 | FSGD2APROT06<br>FSGD2ASPID06 | | |
| | 7 | | W | | FSGD2APROT07<br>FSGD2ASPID07 | | |

Table 28.169 PBG Group (5/8)

| PBG group | PBG channel number | Guard Group | purpose (R or W) | Module to be protected | Guard Register | Error status of PBG group | Note |
|---|---|---|---|---|---|---|---|
| PBG2 | 8 | PG2-SSTM1 | R | • STM1 | FSGD2APROT08<br>FSGD2ASPID08 | ERRSLV2CTL<br>ERRSLV2STAT<br>ERRSLV2TYPE<br>(Included in PG2-PBG2ERR0) | except P1M-C |
| | 9 | | W | | FSGD2APROT09<br>FSGD2ASPID09 | | except P1M-C |
| | 10 | PG2-WDT0 | R | • WDTA0<br>• SWDT | FSGD2APROT10<br>FSGD2ASPID10 | | |
| | 11 | | W | | FSGD2APROT11<br>FSGD2ASPID11 | | |
| | 12 | PG2-WDT1 | R | • WDTA1 | FSGD2APROT12<br>FSGD2ASPID12 | | except P1M-C |
| | 13 | | W | | FSGD2APROT13<br>FSGD2ASPID13 | | except P1M-C |
| | 14 | reserved | | | | | |
| | 15 | reserved | | | | | |
| PBG3 #0 | 0 | PG3-PBG3FIL0 | R | • P-Bus guard control and error register (P-Bus group 3; #0)<br>• P-Bus guard control and error register (P-Bus group 3; #1)<br>• ERRSLV for P-Bus guard 3; #0 & #1 | FSGD3APROT00<br>FSGD3ASPID00 | ERRSLV3CTL<br>ERRSLV3STAT<br>ERRSLV3TYPE<br>(Included in PG3-PBG3ERR0) | |
| | 1 | PG3-PBG3FIL1<br>PG3-PBG3ERR0 | W | | FSGD3APROT01<br>FSGD3ASPID01 | | |
| | 2 | PG3-Startup | R | • ECC on Bus (P-Bus group3)<br>• ECC Controller for Peripheral RAM (MCAN1, TTCAN0,CSIH0/2, FlexRay0-1)<br>• SPID of H-Bus (FlexRay0-1, Ethernet0-1, HS-USRT0-3) | FSGD3APROT02<br>FSGD3ASPID02 | | |
| | 3 | | W | | FSGD3APROT03<br>FSGD3ASPID03 | | |
| | 4 | PG3-RLIN3 | R | • RLIN30/RLIN32 | FSGD3APROT04<br>FSGD3ASPID04 | | |
| | 5 | | W | | FSGD3APROT05<br>FSGD3ASPID05 | | |
| | 6 | PG3-ECM0 | R | • ECM0 | FSGD3APROT06<br>FSGD3ASPID06 | | |
| | 7 | | W | | FSGD3APROT07<br>FSGD3ASPID07 | | |
| | 8 | PG3-one | R | • PIC<br>• DTS merge<br>• DTS trigger selector<br>• ECON1 (FEINT merge for PE1)<br>• ECON2 (FEINT merge for PE2) | FSGD3APROT08<br>FSGD3ASPID08 | | |
| | 9 | | W | | FSGD3APROT09<br>FSGD3ASPID09 | | |
| | 10 | PG3-FLXNTU | R | • FlexRay NTU | FSGD3APROT10<br>FSGD3ASPID10 | | |
| | 11 | | W | | FSGD3APROT11<br>FSGD3ASPID11 | | |
| | 12 | PG3-HBUSG | R | • H-Bus Guard (Ethernet0/1, FlexRay0/1, MEMC) | FSGD3APROT12<br>FSGD3ASPID12 | | |
| | 13 | | W | | FSGD3APROT13<br>FSGD3ASPID13 | | |
| | 14 | PG3-DMHALT | R | • H-Bus DMA Halt | FSGD3APROT14<br>FSGD3ASPID14 | ERRSLV3CTL<br>ERRSLV3STAT<br>ERRSLV3TYPE<br>(Included in PG3-PBG3ERR0) | |
| | 15 | | W | | FSGD3APROT15<br>FSGD3ASPID15 | | |

Table 28.169 PBG Group (6/8)

| PBG group | PBG channel number | Guard Group | purpose (R or W) | Module to be protected | Guard Register | Error status of PBG group | Note |
|---|---|---|---|---|---|---|---|
| PBG3 #1 | 0 | PG3-MCANT | R | • MTTCAN0 | FSGD3BPROT00<br>FSGD3BSPID00 | ERRSLV3CTL<br>ERRSLV3STAT<br>ERRSLV3TYPE<br>(Included in PG3-PBG3ERR0) | |
| | 1 | | W | | FSGD3BPROT01<br>FSGD3BSPID01 | | |
| | 2 | PG3-MCAN1 | R | • MCAN1 | FSGD3BPROT02<br>FSGD3BSPID02 | | |
| | 3 | | W | | FSGD3BPROT03<br>FSGD3BSPID03 | | |
| | 4 | reserved | | | | | |
| | 5 | reserved | | | | | |
| | 6 | PG3-HSUSRT0 | R | • HSUS0 | FSGD3BPROT06<br>FSGD3BSPID06 | | |
| | 7 | | W | | FSGD3BPROT07<br>FSGD3BSPID07 | | |
| | 8 | PG3-HSUSRT2 | R | • HSUS2 | FSGD3BPROT08<br>FSGD3BSPID08 | | except P1M-C, P1H-C (4MB, BGA-156) |
| | 9 | | W | | FSGD3BPROT09<br>FSGD3BSPID09 | | except P1M-C, P1H-C (4MB, BGA-156) |
| | 10 | PG3-CSIH0 | R | • CSIH0 | FSGD3BPROT10<br>FSGD3BSPID10 | | |
| | 11 | | W | | FSGD3BPROT11<br>FSGD3BSPID11 | | |
| | 12 | PG3-CSIH2 | R | • CSIH2 | FSGD3BPROT12<br>FSGD3BSPID12 | | |
| | 13 | | W | | FSGD3BPROT13<br>FSGD3BSPID13 | | |
| | 14 | reserved | | | | | |
| | 15 | reserved | | | | | |

Table 28.169 PBG Group (7/8)

| PBG group | PBG channel number | Guard Group | purpose (R or W) | Module to be protected | Guard Register | Error status of PBG group | Note |
|---|---|---|---|---|---|---|---|
| PBG4 #0 | 0 | PG4-PBG4FIL00<br>PG4-PBG4FIL01<br>PG4-PBG4ERR00 | R | • P-Bus guard control and error register (P-Bus group 4; #0)<br>• P-Bus guard control and error register (P-Bus group 4; #1)<br>• ERRSLV for P-Bus guard 4; #0 & #1 | FSGD4APROT00<br>FSGD4ASPID00 | ERRSLV4CTL<br>ERRSLV4STAT<br>ERRSLV4TYPE<br>(Included in PG4-PBG4ERR00) | |
| | 1 | | W | | FSGD4APROT01<br>FSGD4ASPID01 | | |
| | 2 | PG4-Startup | R | • ECC on Bus (P-Bus group4; ADC0/1 dedicated and others)<br>• Field-BIST Activator<br>• Digital Filter Control (DNFA2 to 7, FLCA0 to 6) | FSGD4APROT02<br>FSGD4ASPID02 | | |
| | 3 | | W | | FSGD4APROT03<br>FSGD4ASPID03 | | |
| | 4 | PG4-one | R | • SINT<br>• ID for Flash | FSGD4APROT04<br>FSGD4ASPID04 | | |
| | 5 | | W | | FSGD4APROT05<br>FSGD4ASPID05 | | |
| | 6 | PG4-ADC0 | R | • ADCF0 | FSGD4APROT06<br>FSGD4ASPID06 | ERRSLV4CTL<br>ERRSLV4STAT<br>ERRSLV4TYPE<br>(Included in PG4-PBG4ERR00) | |
| | 7 | | W | | FSGD4APROT07<br>FSGD4ASPID07 | | |
| | 8 | PG4-ADC1 | R | • ADCF1 | FSGD4APROT08<br>FSGD4ASPID08 | | |
| | 9 | | W | | FSGD4APROT09<br>FSGD4ASPID09 | | |
| | 10 | PG4-SC3 | R | • System Control - Category 3<br>• A group which can be assigned to the ICUMC or to the Safety Core (CKSC0x, CLKD0x) | FSGD4APROT10<br>FSGD4ASPID10 | | |
| | 11 | | W | | FSGD4APROT11<br>FSGD4ASPID11 | | |
| | 12 | PG4-SC2 | R | • System Control - Category 2<br>• A group for the SWDT clock generation | FSGD4APROT12<br>FSGD4ASPID12 | | |
| | 13 | | W | | FSGD4APROT13<br>FSGD4ASPID13 | | |
| | 14 | PG4-SC5 | R | • System Control - Category 5. A group which is assigned to the Safety Core (Cat3 excluding all registers that have an impact on the ICUMC operation e.g. CKSC0x, CLKD0x) *2 | FSGD4APROT14<br>FSGD4ASPID14 | | |
| | 15 | | W | | FSGD4APROT15<br>FSGD4ASPID15 | | |

Table 28.169 PBG Group (8/8)

| PBG group | PBG channel number | Guard Group | purpose (R or W) | Module to be protected | Guard Register | Error status of PBG group | Note |
|---|---|---|---|---|---|---|---|
| PBG4 #1 | 0 | PG4-FLASH | R | • SCDS<br>• FLSCI<br>• FLTM | FSGD4BPROT00<br>FSGD4BSPID00 | ERRSLV4CTL<br>ERRSLV4STAT<br>ERRSLV4TYPE<br>(Included in PG4-PBG4ERR00) | |
| | 1 | | W | | FSGD4BPROT01<br>FSGD4BSPID01 | | |
| | 2 | PG4-FPSYS10 | R | • FACI (Bank A) *2 | FSGD4BPROT02<br>FSGD4BSPID02 | | |
| | 3 | | W | | FSGD4BPROT03<br>FSGD4BSPID03 | | |
| | 4 | PG4-FPSYS11 | R | • FACI (Bank B) *2 | FSGD4BPROT04<br>FSGD4BSPID04 | | |
| | 5 | | W | | FSGD4BPROT05<br>FSGD4BSPID05 | | |
| | 6 | PG4-FLMD1 | R | • FLMD*2 | FSGD4BPROT06<br>FSGD4BSPID06 | | |
| | 7 | | W | | FSGD4BPROT07<br>FSGD4BSPID07 | | |
| | 8 | PG4-PORT | R | • PORT | FSGD4BPROT08<br>FSGD4BSPID08 | | |
| | 9 | | W | | FSGD4BPROT09<br>FSGD4BSPID09 | | |
| | 10 | PG4-SENT | R | • RSENT0 to 9 | FSGD4BPROT10<br>FSGD4BSPID10 | | |
| | 11 | | W | | FSGD4BPROT11<br>FSGD4BSPID11 | | |
| | 12 | PG4-OTS | R | • OTS | FSGD4BPROT12<br>FSGD4BSPID12 | | |
| | 13 | | W | | FSGD4BPROT13<br>FSGD4BSPID13 | | |
| | 14 | reserved | | | | | |
| | 15 | reserved | | | | | |

Note 1. Regarding the PBG registers for the security, see the *Security Hardware Manual*.

Note 2. These guards are available only when ICUMC is disabled. When ICUMC is enabled, an access is guard by PBG4 #2.

### 28.4.3.1 List of Registers

Table 28.170 List of Registers (1/12)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFC4 C000$_H$ | APBFSGDPROT_PDMACM_A | P-bus FS Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | APGRD_PFSS0.PDMCM | **28.4.3.2 (1)** | |
| FFC4 C004$_H$ | APBFSGDSPID_PDMACM_A | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APGRD_PFSS0.PDMCM | **28.4.3.2 (5)** | |
| FFC4 C008$_H$ | APBFSGDPROT_PDMACM_B | P-bus FS Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | APGRD_PFSS0.PDMCM | **28.4.3.2 (2)** | |
| FFC4 C00C$_H$ | APBFSGDSPID_PDMACM_B | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APGRD_PFSS0.PDMCM | **28.4.3.2 (5)** | |
| FFC4 C010$_H$ | APBFSGDPROT_PDMACH_A | P-bus FS Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | APGRD_PFSS0.PDMCH | **28.4.3.2 (1)** | |
| FFC4 C014$_H$ | APBFSGDSPID_PDMACH_A | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APGRD_PFSS0.PDMCH | **28.4.3.2 (5)** | |
| FFC4 C018$_H$ | APBFSGDPROT_PDMACH_B | P-bus FS Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | APGRD_PFSS0.PDMCH | **28.4.3.2 (2)** | |
| FFC4 C01C$_H$ | APBFSGDSPID_PDMACH_B | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APGRD_PFSS0.PDMCH | **28.4.3.2 (5)** | |
| FFC4 C020$_H$ | APBFSGDPROT_INTC2_A | P-bus FS Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | APGRD_PFSS0.INTC2 | **28.4.3.2 (1)** | |
| FFC4 C024$_H$ | APBFSGDSPID_INTC2_A | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APGRD_PFSS0.INTC2 | **28.4.3.2 (5)** | |
| FFC4 C028$_H$ | APBFSGDPROT_INTC2_B | P-bus FS Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | APGRD_PFSS0.INTC2 | **28.4.3.2 (2)** | |
| FFC4 C02C$_H$ | APBFSGDSPID_INTC2_B | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APGRD_PFSS0.INTC2 | **28.4.3.2 (5)** | |
| FFC4 C040$_H$ | APBFSGDPROT_PBG_A | P-bus FS Guard Protection Setting Register | R/W | 064D FE1B$_H$ | 8/16/32 | APBGRD_PFSS1.PBG | **28.4.3.2 (3)** | |
| FFC4 C044$_H$ | APBFSGDSPID_PBG_A | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APBGRD_PFSS1.PBG | **28.4.3.2 (5)** | |
| FFC4 C048$_H$ | APBFSGDPROT_PBG_B | P-bus FS Guard Protection Setting Register | R/W | 064D FE17$_H$ | 8/16/32 | APBGRD_PFSS1.PBG | **28.4.3.2 (4)** | |
| FFC4 C04C$_H$ | APBFSGDSPID_PBG_B | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APBGRD_PFSS1.PBG | **28.4.3.2 (5)** | |
| FFC4 C050$_H$ | APBFSGDPROT_SP1_A | P-bus FS Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | APBGRD_PFSS1.SP1 | **28.4.3.2 (1)** | |
| FFC4 C054$_H$ | APBFSGDSPID_SP1_A | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APBGRD_PFSS1.SP1 | **28.4.3.2 (5)** | |
| FFC4 C058$_H$ | APBFSGDPROT_SP1_B | P-bus FS Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | APBGRD_PFSS1.SP1 | **28.4.3.2 (2)** | |
| FFC4 C05C$_H$ | APBFSGDSPID_SP1_B | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APBGRD_PFSS1.SP1 | **28.4.3.2 (5)** | |
| FFC4 C060$_H$ | APBFSGDPROT_SP2_A | P-bus FS Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | APBGRD_PFSS1.SP2 | **28.4.3.2 (1)** | except P1M-C |
| FFC4 C064$_H$ | APBFSGDSPID_SP2_A | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APBGRD_PFSS1.SP2 | **28.4.3.2 (5)** | except P1M-C |
| FFC4 C068$_H$ | APBFSGDPROT_SP2_B | P-bus FS Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | APBGRD_PFSS1.SP2 | **28.4.3.2 (2)** | except P1M-C |
| FFC4 C06C$_H$ | APBFSGDSPID_SP2_B | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APBGRD_PFSS1.SP2 | **28.4.3.2 (5)** | except P1M-C |

Table 28.170 List of Registers (2/12)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFC4 C070$_H$ | APBFSGDPROT_SP4_A | P-bus FS Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.3.2 (1)** | |
| FFC4 C074$_H$ | APBFSGDSPID_SP4_A | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.3.2 (5)** | |
| FFC4 C078$_H$ | APBFSGDPROT_SP4_B | P-bus FS Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.3.2 (2)** | |
| FFC4 C07C$_H$ | APBFSGDSPID_SP4_B | P-bus FS Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | APBGRD_PFSS1.SP4 | **28.4.3.2 (5)** | |
| FFC4 C800$_H$ | ERRSLVCTL_PFSS0 | ERRSLV Control Register for PFSS P-bus FS Guard Slave0 | W | 0000 0000$_H$ | 32 | — | **28.4.3.2 (6)** | Included in APBGRD_PFSS1.SP4 |
| FFC4 C804$_H$ | ERRSLVSTAT_PFSS0 | ERRSLV Status Register for PFSS P-bus FS Guard Slave0 | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (7)** | Included in APBGRD_PFSS1.SP4 |
| FFC4 C80C$_H$ | ERRSLVTYPE_PFSS0 | ERRSLV Error Transfer Type Register for PFSS P-bus FS Guard Slave0 | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (8)** | Included in APBGRD_PFSS1.SP4 |
| FFC4 C810$_H$ | ERRSLVCTL_PFSS1 | ERRSLV Control Register for PFSS P-bus FS Guard Slave1 | W | 0000 0000$_H$ | 32 | — | **28.4.3.2 (6)** | Included in APBGRD_PFSS1.SP4 |
| FFC4 C814$_H$ | ERRSLVSTAT_PFSS1 | ERRSLV Status Register for PFSS P-bus FS Guard Slave1 | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (7)** | Included in APBGRD_PFSS1.SP4 |
| FFC4 C81C$_H$ | ERRSLVTYPE_PFSS1 | ERRSLV Error Transfer Type Register for PFSS P-bus FS Guard Slave1 | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (8)** | Included in APBGRD_PFSS1.SP4 |
| FFC5 A100$_H$ | ERRSLV5ACTL | ERRSLV Control Register for P-bus Guard | W | 00$_H$ | 8 | — | **28.4.3.2 (11)** | Included in PG5-PBG5ERR0 |
| FFC5 A104$_H$ | ERRSLV5ASTAT | ERRSLV Status Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (12)** | Included in PG5-PBG5ERR0 |
| FFC5 A10C$_H$ | ERRSLV5ATYPE | ERRSLV Error Transfer Type Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (13)** | Included in PG5-PBG5ERR0 |
| FFC5 A300$_H$ | ERRSLV5BCTL | ERRSLV Control Register for P-bus Guard | W | 00$_H$ | 8 | — | **28.4.3.2 (11)** | Included in PG5-PBG5ERR1 |
| FFC5 A304$_H$ | ERRSLV5BSTAT | ERRSLV Status Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (12)** | Included in PG5-PBG5ERR1 |
| FFC5 A30C$_H$ | ERRSLV5BTYPE | ERRSLV Error Transfer Type Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (13)** | Included in PG5-PBG5ERR1 |
| FFEE 0000$_H$ | FSGD1APROT00 | P-Bus Guard Protection Setting Register | R/W | 064D FE1B$_H$ | 8/16/32 | PBG1#0.PG1-PBG1FIL0 PBG1#0.PG1-PBG1FIL1 PBG1#0.PG1-PBG1ERR0 PBG1#0.PG1-PBG1ERRI | **28.4.3.2 (9)** | |

Table 28.170 List of Registers (3/12)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFEE 0004$_H$ | FSGD1ASPID00 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#0.PG1-PBG1FIL0 PBG1#0.PG1-PBG1FIL1 PBG1#0.PG1-PBG1ERR0 PBG1#0.PG1-PBG1ERRI | **28.4.3.2 (10)** | |
| FFEE 0008$_H$ | FSGD1APROT01 | P-Bus Guard Protection Setting Register | R/W | 064D FE17$_H$ | 8/16/32 | PBG1#0.PG1-PBG1FIL0 PBG1#0.PG1-PBG1FIL1 PBG1#0.PG1-PBG1ERR0 PBG1#0.PG1-PBG1ERRI | **28.4.3.2 (9)** | |
| FFEE 000C$_H$ | FSGD1ASPID01 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#0.PG1-PBG1FIL0 PBG1#0.PG1-PBG1FIL1 PBG1#0.PG1-PBG1ERR0 PBG1#0.PG1-PBG1ERRI | **28.4.3.2 (10)** | |
| FFEE 0010$_H$ | FSGD1APROT02 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG1#0.PG1-Startup | **28.4.3.2 (9)** | |
| FFEE 0014$_H$ | FSGD1ASPID02 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#0.PG1-Startup | **28.4.3.2 (10)** | |
| FFEE 0018$_H$ | FSGD1APROT03 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG1#0.PG1-Startup | **28.4.3.2 (9)** | |
| FFEE 001C$_H$ | FSGD1ASPID03 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#0.PG1-Startup | **28.4.3.2 (10)** | |
| FFEE 0020$_H$ | FSGD1APROT04 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG1#0.PG1-RLIN3 | **28.4.3.2 (9)** | |
| FFEE 0024$_H$ | FSGD1ASPID04 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#0.PG1-RLIN3 | **28.4.3.2 (10)** | |
| FFEE 0028$_H$ | FSGD1APROT05 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG1#0.PG1-RLIN3 | **28.4.3.2 (9)** | |
| FFEE 002C$_H$ | FSGD1ASPID05 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#0.PG1-RLIN3 | **28.4.3.2 (10)** | |
| FFEE 0030$_H$ | FSGD1APROT06 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG1#0.PG1-ECM1 | **28.4.3.2 (9)** | except P1M-C |
| FFEE 0034$_H$ | FSGD1ASPID06 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#0.PG1-ECM1 | **28.4.3.2 (10)** | except P1M-C |
| FFEE 0038$_H$ | FSGD1APROT07 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG1#0.PG1-ECM1 | **28.4.3.2 (9)** | except P1M-C |
| FFEE 003C$_H$ | FSGD1ASPID07 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#0.PG1-ECM1 | **28.4.3.2 (10)** | except P1M-C |
| FFEE 0100$_H$ | ERRSLV1CTL | ERRSLV Control Register for P-bus Guard | W | 00$_H$ | 8 | — | **28.4.3.2 (11)** | Included in PBG1#0.PG1-PBG1ERR0 |
| FFEE 0104$_H$ | ERRSLV1STAT | ERRSLV Status Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (12)** | Included in PBG1#0.PG1-PBG1ERR0 |
| FFEE 010C$_H$ | ERRSLV1TYPE | ERRSLV Error Transfer Type Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (13)** | Included in PBG1#0.PG1-PBG1ERR0 |

Table 28.170 List of Registers (4/12)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFEE 0200$_H$ | FSGD1BPROT00 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG1#1.PG1-MCAN0 | **28.4.3.2 (9)** | |
| FFEE 0204$_H$ | FSGD1BSPID00 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-MCAN0 | **28.4.3.2 (10)** | |
| FFEE 0208$_H$ | FSGD1BPROT01 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG1#1.PG1-MCAN0 | **28.4.3.2 (9)** | |
| FFEE 020C$_H$ | FSGD1BSPID01 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-MCAN0 | **28.4.3.2 (10)** | |
| FFEE 0210$_H$ | FSGD1BPROT02 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG1#1.PG1-MCAN2 | **28.4.3.2 (9)** | except P1M-C, P1H-C (4MB) |
| FFEE 0214$_H$ | FSGD1BSPID02 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-MCAN2 | **28.4.3.2 (10)** | except P1M-C, P1H-C (4MB) |
| FFEE 0218$_H$ | FSGD1BPROT03 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG1#1.PG1-MCAN2 | **28.4.3.2 (9)** | except P1M-C, P1H-C (4MB) |
| FFEE 021C$_H$ | FSGD1BSPID03 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-MCAN2 | **28.4.3.2 (10)** | except P1M-C, P1H-C (4MB) |
| FFEE 0230$_H$ | FSGD1BPROT06 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG1#1.PG1-HSUSRT1 | **28.4.3.2 (9)** | |
| FFEE 0234$_H$ | FSGD1BSPID06 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-HSUSRT1 | **28.4.3.2 (10)** | |
| FFEE 0238$_H$ | FSGD1BPROT07 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG1#1.PG1-HSUSRT1 | **28.4.3.2 (9)** | |
| FFEE 023C$_H$ | FSGD1BSPID07 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-HSUSRT1 | **28.4.3.2 (10)** | |
| FFEE 0240$_H$ | FSGD1BPROT08 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG1#1.PG1-HSUSRT3 | **28.4.3.2 (9)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFEE 0244$_H$ | FSGD1BSPID08 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-HSUSRT3 | **28.4.3.2 (10)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFEE 0248$_H$ | FSGD1BPROT09 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG1#1.PG1-HSUSRT3 | **28.4.3.2 (9)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFEE 024C$_H$ | FSGD1BSPID09 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-HSUSRT3 | **28.4.3.2 (10)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFEE 0250$_H$ | FSGD1BPROT10 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG1#1.PG1-CSIH1 | **28.4.3.2 (9)** | |
| FFEE 0254$_H$ | FSGD1BSPID10 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-CSIH1 | **28.4.3.2 (10)** | |
| FFEE 0258$_H$ | FSGD1BPROT11 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG1#1.PG1-CSIH1 | **28.4.3.2 (9)** | |
| FFEE 025C$_H$ | FSGD1BSPID11 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-CSIH1 | **28.4.3.2 (10)** | |

Table 28.170 List of Registers (5/12)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFEE 0260$_H$ | FSGD1BPROT12 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG1#1.PG1-CSIH3 | **28.4.3.2 (9)** | |
| FFEE 0264$_H$ | FSGD1BSPID12 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-CSIH3 | **28.4.3.2 (10)** | |
| FFEE 0268$_H$ | FSGD1BPROT13 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG1#1.PG1-CSIH3 | **28.4.3.2 (9)** | |
| FFEE 026C$_H$ | FSGD1BSPID13 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG1#1.PG1-CSIH3 | **28.4.3.2 (10)** | |
| FFDD D000$_H$ | FSGD2APROT00 | P-Bus Guard Protection Setting Register | R/W | 064D FE1B$_H$ | 8/16/32 | PBG2.PG2-PBG2FIL PBG2.PG2-PBG2ERR0 PBG2.PG2-PBG2ERRI | **28.4.3.2 (9)** | |
| FFDD D004$_H$ | FSGD2ASPID00 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-PBG2FIL PBG2.PG2-PBG2ERR0 PBG2.PG2-PBG2ERRI | **28.4.3.2 (10)** | |
| FFDD D008$_H$ | FSGD2APROT01 | P-Bus Guard Protection Setting Register | R/W | 064D FE17$_H$ | 8/16/32 | PBG2.PG2-PBG2FIL PBG2.PG2-PBG2ERR0 PBG2.PG2-PBG2ERRI | **28.4.3.2 (9)** | |
| FFDD D00C$_H$ | FSGD2ASPID01 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-PBG2FIL PBG2.PG2-PBG2ERR0 PBG2.PG2-PBG2ERRI | **28.4.3.2 (10)** | |
| FFDD D010$_H$ | FSGD2APROT02 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG2.PG2-Startup | **28.4.3.2 (9)** | |
| FFDD D014$_H$ | FSGD2ASPID02 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-Startup | **28.4.3.2 (10)** | |
| FFDD D018$_H$ | FSGD2APROT03 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG2.PG2-Startup | **28.4.3.2 (9)** | |
| FFDD D01C$_H$ | FSGD2ASPID03 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-Startup | **28.4.3.2 (10)** | |
| FFDD D020$_H$ | FSGD2APROT04 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG2.PG2-GTM0 | **28.4.3.2 (9)** | |
| FFDD D024$_H$ | FSGD2ASPID04 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-GTM0 | **28.4.3.2 (10)** | |
| FFDD D028$_H$ | FSGD2APROT05 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG2.PG2-GTM0 | **28.4.3.2 (9)** | |
| FFDD D02C$_H$ | FSGD2ASPID05 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-GTM0 | **28.4.3.2 (10)** | |
| FFDD D030$_H$ | FSGD2APROT06 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG2.PG2-SSTM0 | **28.4.3.2 (9)** | |
| FFDD D034$_H$ | FSGD2ASPID06 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-SSTM0 | **28.4.3.2 (10)** | |
| FFDD D038$_H$ | FSGD2APROT07 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG2.PG2-SSTM0 | **28.4.3.2 (9)** | |
| FFDD D03C$_H$ | FSGD2ASPID07 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-SSTM0 | **28.4.3.2 (10)** | |
| FFDD D040$_H$ | FSGD2APROT08 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG2.PG2-SSTM1 | **28.4.3.2 (9)** | except P1M-C |
| FFDD D044$_H$ | FSGD2ASPID08 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-SSTM1 | **28.4.3.2 (10)** | except P1M-C |

Table 28.170 List of Registers (6/12)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFDD D048$_H$ | FSGD2APROT09 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG2.PG2-SSTM1 | **28.4.3.2 (9)** | except P1M-C |
| FFDD D04C$_H$ | FSGD2ASPID09 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-SSTM1 | **28.4.3.2 (10)** | except P1M-C |
| FFDD D050$_H$ | FSGD2APROT10 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG2.PG2-WDT0 | **28.4.3.2 (9)** | |
| FFDD D054$_H$ | FSGD2ASPID10 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-WDT0 | **28.4.3.2 (10)** | |
| FFDD D058$_H$ | FSGD2APROT11 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG2.PG2-WDT0 | **28.4.3.2 (9)** | |
| FFDD D05C$_H$ | FSGD2ASPID11 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-WDT0 | **28.4.3.2 (10)** | |
| FFDD D060$_H$ | FSGD2APROT12 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG2.PG2-WDT1 | **28.4.3.2 (9)** | except P1M-C |
| FFDD D064$_H$ | FSGD2ASPID12 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-WDT1 | **28.4.3.2 (10)** | except P1M-C |
| FFDD D068$_H$ | FSGD2APROT13 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG2.PG2-WDT1 | **28.4.3.2 (9)** | except P1M-C |
| FFDD D06C$_H$ | FSGD2ASPID13 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG2.PG2-WDT1 | **28.4.3.2 (10)** | except P1M-C |
| FFDD D100$_H$ | ERRSLV2CTL | ERRSLV Control Register for P-bus Guard | W | 00$_H$ | 8 | — | **28.4.3.2 (11)** | Included in PBG2.PG2-PBG2ERR0 |
| FFDD D104$_H$ | ERRSLV2STAT | ERRSLV Status Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (12)** | Included in PBG2.PG2-PBG2ERR0 |
| FFDD D10C$_H$ | ERRSLV2TYPE | ERRSLV Error Transfer Type Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (13)** | Included in PBG2.PG2-PBG2ERR0 |
| FFF9 4000$_H$ | FSGD3APROT00 | P-Bus Guard Protection Setting Register | R/W | 064D FE1B$_H$ | 8/16/32 | PBG3#0.PG3-PBG3FIL0 PBG3#0.PG3-PBG3FIL1 PBG3#0.PG3-PBG3ERR0 PBG3#0.PG3-PBG3ERRI | **28.4.3.2 (9)** | |
| FFF9 4004$_H$ | FSGD3ASPID00 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-PBG3FIL0 PBG3#0.PG3-PBG3FIL1 PBG3#0.PG3-PBG3ERR0 PBG3#0.PG3-PBG3ERRI | **28.4.3.2 (10)** | |
| FFF9 4008$_H$ | FSGD3APROT01 | P-Bus Guard Protection Setting Register | R/W | 064D FE17$_H$ | 8/16/32 | PBG3#0.PG3-PBG3FIL0 PBG3#0.PG3-PBG3FIL1 PBG3#0.PG3-PBG3ERR0 PBG3#0.PG3-PBG3ERRI | **28.4.3.2 (9)** | |

Table 28.170 List of Registers (7/12)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFF9 400C$_H$ | FSGD3ASPID01 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-PBG3FIL0 PBG3#0.PG3-PBG3FIL1 PBG3#0.PG3-PBG3ERR0 PBG3#0.PG3-PBG3ERRI | **28.4.3.2 (10)** | |
| FFF9 4010$_H$ | FSGD3APROT02 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#0.PG3-Startup | **28.4.3.2 (9)** | |
| FFF9 4014$_H$ | FSGD3ASPID02 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-Startup | **28.4.3.2 (10)** | |
| FFF9 4018$_H$ | FSGD3APROT03 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#0.PG3-Startup | **28.4.3.2 (9)** | |
| FFF9 401C$_H$ | FSGD3ASPID03 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-Startup | **28.4.3.2 (10)** | |
| FFF9 4020$_H$ | FSGD3APROT04 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#0.PG3-RLIN3 | **28.4.3.2 (9)** | |
| FFF9 4024$_H$ | FSGD3ASPID04 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-RLIN3 | **28.4.3.2 (10)** | |
| FFF9 4028$_H$ | FSGD3APROT05 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#0.PG3-RLIN3 | **28.4.3.2 (9)** | |
| FFF9 402C$_H$ | FSGD3ASPID05 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-RLIN3 | **28.4.3.2 (10)** | |
| FFF9 4030$_H$ | FSGD3APROT06 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#0.PG3-ECM0 | **28.4.3.2 (9)** | |
| FFF9 4034$_H$ | FSGD3ASPID06 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-ECM0 | **28.4.3.2 (10)** | |
| FFF9 4038$_H$ | FSGD3APROT07 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#0.PG3-ECM0 | **28.4.3.2 (9)** | |
| FFF9 403C$_H$ | FSGD3ASPID07 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-ECM0 | **28.4.3.2 (10)** | |
| FFF9 4040$_H$ | FSGD3APROT08 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#0.PG3-one | **28.4.3.2 (9)** | |
| FFF9 4044$_H$ | FSGD3ASPID08 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-one | **28.4.3.2 (10)** | |
| FFF9 4048$_H$ | FSGD3APROT09 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#0.PG3-one | **28.4.3.2 (9)** | |
| FFF9 404C$_H$ | FSGD3ASPID09 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-one | **28.4.3.2 (10)** | |
| FFF9 4050$_H$ | FSGD3APROT10 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#0.PG3-FLXNTU | **28.4.3.2 (9)** | |
| FFF9 4054$_H$ | FSGD3ASPID10 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-FLXNTU | **28.4.3.2 (10)** | |
| FFF9 4058$_H$ | FSGD3APROT11 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#0.PG3-FLXNTU | **28.4.3.2 (9)** | |
| FFF9 405C$_H$ | FSGD3ASPID11 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-FLXNTU | **28.4.3.2 (10)** | |
| FFF9 4060$_H$ | FSGD3APROT12 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#0.PG3-HBUSG | **28.4.3.2 (9)** | |
| FFF9 4064$_H$ | FSGD3ASPID12 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-HBUSG | **28.4.3.2 (10)** | |

Table 28.170 List of Registers (8/12)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFF9 4068$_H$ | FSGD3APROT13 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#0.PG3-HBUSG | **28.4.3.2 (9)** | |
| FFF9 406C$_H$ | FSGD3ASPID13 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-HBUSG | **28.4.3.2 (10)** | |
| FFF9 4070$_H$ | FSGD3APROT14 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#0.PG3-DMHALT | **28.4.3.2 (9)** | |
| FFF9 4074$_H$ | FSGD3ASPID14 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-DMHALT | **28.4.3.2 (10)** | |
| FFF9 4078$_H$ | FSGD3APROT15 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#0.PG3-DMHALT | **28.4.3.2 (9)** | |
| FFF9 407C$_H$ | FSGD3ASPID15 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#0.PG3-DMHALT | **28.4.3.2 (10)** | |
| FFF9 4100$_H$ | ERRSLV3CTL | ERRSLV Control Register for P-bus Guard | W | 00$_H$ | 8 | — | **28.4.3.2 (11)** | Included in PBG3#0.PG3-PBG3ERR0 |
| FFF9 4104$_H$ | ERRSLV3STAT | ERRSLV Status Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (12)** | Included in PBG3#0.PG3-PBG3ERR0 |
| FFF9 410C$_H$ | ERRSLV3TYPE | ERRSLV Error Transfer Type Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (13)** | Included in PBG3#0.PG3-PBG3ERR0 |
| FFF94200$_H$ | FSGD3BPROT00 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#1.PG3-MCANT | **28.4.3.2 (9)** | |
| FFF9 4204$_H$ | FSGD3BSPID00 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-MCANT | **28.4.3.2 (10)** | |
| FFF9 4208$_H$ | FSGD3BPROT01 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#1.PG3-MCANT | **28.4.3.2 (9)** | |
| FFF9 420C$_H$ | FSGD3BSPID01 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-MCANT | **28.4.3.2 (10)** | |
| FFF9 4210$_H$ | FSGD3BPROT02 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#1.PG3-MCAN1 | **28.4.3.2 (9)** | |
| FFF9 4214$_H$ | FSGD3BSPID02 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-MCAN1 | **28.4.3.2 (10)** | |
| FFF9 4218$_H$ | FSGD3BPROT03 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#1.PG3-MCAN1 | **28.4.3.2 (9)** | |
| FFF9 421C$_H$ | FSGD3BSPID03 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-MCAN1 | **28.4.3.2 (10)** | |
| FFF9 4230$_H$ | FSGD3BPROT06 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#1.PG3-HSUSRT0 | **28.4.3.2 (9)** | |
| FFF9 4234$_H$ | FSGD3BSPID06 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-HSUSRT0 | **28.4.3.2 (10)** | |
| FFF9 4238$_H$ | FSGD3BPROT07 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#1.PG3-HSUSRT0 | **28.4.3.2 (9)** | |
| FFF9 423C$_H$ | FSGD3BSPID07 | P-Bus Guard Protection Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-HSUSRT0 | **28.4.3.2 (10)** | |
| FFF9 4240$_H$ | FSGD3BPROT08 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#1.PG3-HSUSRT2 | **28.4.3.2 (9)** | except P1M-C, P1H-C (4MB, BGA-156) |

Table 28.170 List of Registers (9/12)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFF9 4244$_H$ | FSGD3BSPID08 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-HSUSRT2 | **28.4.3.2 (10)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFF9 4248$_H$ | FSGD3BPROT09 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#1.PG3-HSUSRT2 | **28.4.3.2 (9)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFF9 424C$_H$ | FSGD3BSPID09 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-HSUSRT2 | **28.4.3.2 (10)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFF9 4250$_H$ | FSGD3BPROT10 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#1.PG3-CSIH0 | **28.4.3.2 (9)** | |
| FFF9 4254$_H$ | FSGD3BSPID10 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-CSIH0 | **28.4.3.2 (10)** | |
| FFF9 4258$_H$ | FSGD3BPROT11 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#1.PG3-CSIH0 | **28.4.3.2 (9)** | |
| FFF9 425C$_H$ | FSGD3BSPID11 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-CSIH0 | **28.4.3.2 (10)** | |
| FFF9 4260$_H$ | FSGD3BPROT12 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG3#1.PG3-CSIH2 | **28.4.3.2 (9)** | |
| FFF9 4264$_H$ | FSGD3BSPID12 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-CSIH2 | **28.4.3.2 (10)** | |
| FFF9 4268$_H$ | FSGD3BPROT13 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG3#1.PG3-CSIH2 | **28.4.3.2 (9)** | |
| FFF9 426C$_H$ | FSGD3BSPID13 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG3#1.PG3-CSIH2 | **28.4.3.2 (10)** | |
| FFF9 0000$_H$ | FSGD4APROT00 | P-Bus Guard Protection Setting Register | R/W | 064D FE1B$_H$ | 8/16/32 | PBG4#0.PG4-PBG4FIL00 PBG4#0.PG4-PBG4FIL01 PBG4#0.PG4-PBG4ERR00 PBG4#0.PG4-PBG4ERRI0 | **28.4.3.2 (9)** | |
| FFF9 0004$_H$ | FSGD4ASPID00 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-PBG4FIL00 PBG4#0.PG4-PBG4FIL01 PBG4#0.PG4-PBG4ERR00 PBG4#0.PG4-PBG4ERRI0 | **28.4.3.2 (10)** | |
| FFF9 0008$_H$ | FSGD4APROT01 | P-Bus Guard Protection Setting Register | R/W | 064D FE17$_H$ | 8/16/32 | PBG4#0.PG4-PBG4FIL00 PBG4#0.PG4-PBG4FIL01 PBG4#0.PG4-PBG4ERR00 PBG4#0.PG4-PBG4ERRI0 | **28.4.3.2 (9)** | |
| FFF9 000C$_H$ | FSGD4ASPID01 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-PBG4FIL00 PBG4#0.PG4-PBG4FIL01 PBG4#0.PG4-PBG4ERR00 PBG4#0.PG4-PBG4ERRI0 | **28.4.3.2 (10)** | |
| FFF9 0010$_H$ | FSGD4APROT02 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#0.PG4-Startup | **28.4.3.2 (9)** | |

Table 28.170 List of Registers (10/12)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFF9 0014$_H$ | FSGD4ASPID02 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-Startup | **28.4.3.2 (10)** | |
| FFF9 0018$_H$ | FSGD4APROT03 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#0.PG4-Startup | **28.4.3.2 (9)** | |
| FFF9 001C$_H$ | FSGD4ASPID03 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-Startup | **28.4.3.2 (10)** | |
| FFF9 0020$_H$ | FSGD4APROT04 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#0.PG4-one | **28.4.3.2 (9)** | |
| FFF9 0024$_H$ | FSGD4ASPID04 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-one | **28.4.3.2 (10)** | |
| FFF9 0028$_H$ | FSGD4APROT05 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#0.PG4-one | **28.4.3.2 (9)** | |
| FFF9 002C$_H$ | FSGD4ASPID05 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-one | **28.4.3.2 (10)** | |
| FFF9 0030$_H$ | FSGD4APROT06 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#0.PG4-ADC0 | **28.4.3.2 (9)** | |
| FFF9 0034$_H$ | FSGD4ASPID06 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-ADC0 | **28.4.3.2 (10)** | |
| FFF9 0038$_H$ | FSGD4APROT07 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#0.PG4-ADC0 | **28.4.3.2 (9)** | |
| FFF9 003C$_H$ | FSGD4ASPID07 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-ADC0 | **28.4.3.2 (10)** | |
| FFF9 0040$_H$ | FSGD4APROT08 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#0.PG4-ADC1 | **28.4.3.2 (9)** | |
| FFF9 0044$_H$ | FSGD4ASPID08 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-ADC1 | **28.4.3.2 (10)** | |
| FFF9 0048$_H$ | FSGD4APROT09 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#0.PG4-ADC1 | **28.4.3.2 (9)** | |
| FFF9 004C$_H$ | FSGD4ASPID09 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-ADC1 | **28.4.3.2 (10)** | |
| FFF9 0050$_H$ | FSGD4APROT10 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#0.PG4-SC3 | **28.4.3.2 (9)** | |
| FFF9 0054$_H$ | FSGD4ASPID10 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-SC3 | **28.4.3.2 (10)** | |
| FFF9 0058$_H$ | FSGD4APROT11 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#0.PG4-SC3 | **28.4.3.2 (9)** | |
| FFF9 005C$_H$ | FSGD4ASPID11 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-SC3 | **28.4.3.2 (10)** | |
| FFF9 0060$_H$ | FSGD4APROT12 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#0.PG4-SC2 | **28.4.3.2 (9)** | |
| FFF9 0064$_H$ | FSGD4ASPID12 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-SC2 | **28.4.3.2 (10)** | |
| FFF9 0068$_H$ | FSGD4APROT13 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#0.PG4-SC2 | **28.4.3.2 (9)** | |
| FFF9 006C$_H$ | FSGD4ASPID13 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-SC2 | **28.4.3.2 (10)** | |
| FFF9 0070$_H$ | FSGD4APROT14 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#0.PG4-SC5 | **28.4.3.2 (9)** | |
| FFF9 0074$_H$ | FSGD4ASPID14 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-SC5 | **28.4.3.2 (10)** | |

Table 28.170 List of Registers (11/12)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFF9 0078$_H$ | FSGD4APROT15 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#0.PG4-SC5 | **28.4.3.2 (9)** | |
| FFF9 007C$_H$ | FSGD4ASPID15 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#0.PG4-SC5 | **28.4.3.2 (10)** | |
| FFF9 0100$_H$ | ERRSLV4CTL | ERRSLV Control Register for P-bus Guard | W | 00$_H$ | 8 | — | **28.4.3.2 (11)** | Included in PBG4#0.PG4-PBG4ERR00 |
| FFF9 0104$_H$ | ERRSLV4STAT | ERRSLV Status Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (12)** | Included in PBG4#0.PG4-PBG4ERR00 |
| FFF9 010C$_H$ | ERRSLV4TYPE | ERRSLV Error Transfer Type Register for P-Bus Guard | R | 0000 0000$_H$ | 32 | — | **28.4.3.2 (13)** | Included in PBG4#0.PG4-PBG4ERR00 |
| FFF9 0200$_H$ | FSGD4BPROT00 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#1.PG4-FLASH | **28.4.3.2 (9)** | |
| FFF9 0204$_H$ | FSGD4BSPID00 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-FLASH | **28.4.3.2 (10)** | |
| FFF9 0208$_H$ | FSGD4BPROT01 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#1.PG4-FLASH | **28.4.3.2 (9)** | |
| FFF9 020C$_H$ | FSGD4BSPID01 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-FLASH | **28.4.3.2 (10)** | |
| FFF9 0210$_H$ | FSGD4BPROT02 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#1.PG4-FPSYS10 | **28.4.3.2 (9)** | |
| FFF9 0214$_H$ | FSGD4BSPID02 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-FPSYS10 | **28.4.3.2 (10)** | |
| FFF9 0218$_H$ | FSGD4BPROT03 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#1.PG4-FPSYS10 | **28.4.3.2 (9)** | |
| FFF9 021C$_H$ | FSGD4BSPID03 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-FPSYS10 | **28.4.3.2 (10)** | |
| FFF9 0220$_H$ | FSGD4BPROT04 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#1.PG4-FPSYS11 | **28.4.3.2 (9)** | |
| FFF9 0224$_H$ | FSGD4BSPID04 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-FPSYS11 | **28.4.3.2 (10)** | |
| FFF9 0228$_H$ | FSGD4BPROT05 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#1.PG4-FPSYS11 | **28.4.3.2 (9)** | |
| FFF9 022C$_H$ | FSGD4BSPID05 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-FPSYS11 | **28.4.3.2 (10)** | |
| FFF9 0230$_H$ | FSGD4BPROT06 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#1.PG4-FLMD1 | **28.4.3.2 (9)** | |
| FFF9 0234$_H$ | FSGD4BSPID06 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-FLMD1 | **28.4.3.2 (10)** | |
| FFF9 0238$_H$ | FSGD4BPROT07 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#1.PG4-FLMD1 | **28.4.3.2 (9)** | |
| FFF9 023C$_H$ | FSGD4BSPID07 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-FLMD1 | **28.4.3.2 (10)** | |
| FFF9 0240$_H$ | FSGD4BPROT08 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#1.PG4-PORT | **28.4.3.2 (9)** | |
| FFF9 0244$_H$ | FSGD4BSPID08 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-PORT | **28.4.3.2 (10)** | |

**Table 28.170 List of Registers (12/12)**

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFF9 0248$_H$ | FSGD4BPROT09 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#1.PG4-PORT | **28.4.3.2 (9)** | |
| FFF9 024C$_H$ | FSGD4BSPID09 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-PORT | **28.4.3.2 (10)** | |
| FFF9 0250$_H$ | FSGD4BPROT10 | P-Bus Guard Protection Setting Register | R/W | 060D FE1B$_H$ | 8/16/32 | PBG4#1.PG4-SENT | **28.4.3.2 (9)** | |
| FFF9 0254$_H$ | FSGD4BSPID10 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-SENT | **28.4.3.2 (10)** | |
| FFF9 0258$_H$ | FSGD4BPROT11 | P-Bus Guard Protection Setting Register | R/W | 060D FE17$_H$ | 8/16/32 | PBG4#1.PG4-SENT | **28.4.3.2 (9)** | |
| FFF9 025C$_H$ | FSGD4BSPID11 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-SENT | **28.4.3.2 (10)** | |
| FFF9 0260$_H$ | FSGD4BPROT12 | P-Bus Guard Protection Setting Register | R/W | 064D FE1B$_H$ | 8/16/32 | PBG4#1.PG4-OTS | **28.4.3.2 (9)** | |
| FFF9 0264$_H$ | FSGD4BSPID12 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-OTS | **28.4.3.2 (10)** | |
| FFF9 0268$_H$ | FSGD4BPROT13 | P-Bus Guard Protection Setting Register | R/W | 064D FE17$_H$ | 8/16/32 | PBG4#1.PG4-OTS | **28.4.3.2 (9)** | |
| FFF9 026C$_H$ | FSGD4BSPID13 | P-bus Guard SPID Setting Register | R/W | FFFF FFFF$_H$ | 8/16/32 | PBG4#1.PG4-OTS | **28.4.3.2 (10)** | |

### 28.4.3.2 Details of Registers

#### (1) APBFSGDPROT_PDMACM_A/PDMACH_A/INTC2_A/SP*n*_A — P-bus FS Guard Protection Setting Register (*n* = 1, 2, 4)

These registers are setting for P-bus guard at read cycle.

**Access:** These registers can be Read/written in 32/16/8-bit units.

**Address:** APBFSGDPROT_PDMACM_A: FFC4 C000$_H$
APBFSGDPROT_PDMACH_A: FFC4 C010$_H$
APBFSGDPROT_INTC2_A: FFC4 C020$_H$
APBFSGDPROT_SP1_A: FFC4 C050$_H$
APBFSGDPROT_SP2_A: FFC4 C060$_H$
APBFSGDPROT_SP4_A: FFC4 C070$_H$

**Value after reset:** 060D FE1B$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOCK | — | — | — | — | — | — | — | — | PEID5 | PEID4 | — | PEID2 | PEID1 | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.171 APBFSGDPROT_PDMACM_A/PDMACH_A/INTC2_A/SP*n*_A register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | LOCK | Lock Bit<br>0: Registers can be-written<br>1: Any further write to APBFSGDPROT_* and APBFSGDSPID_* are ignored.<br>This bit can only be cleared by reset. |
| 30 to 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22, 21 | PEID[5:4] | Access with PEID<br>PEID is a bit list where each bit represents one PEID value. Setting more than one bit allows to enable more than one virtual Processor ID value at a time.<br>0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 20 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 19, 18 | PEID[2:1] | Access with PEID<br>PEID is a bit list where each bit represents one PEID value. Setting more than one bit allows to enable more than one virtual Processor ID value at a time.<br>0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 17 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (2) APBFSGDPROT_PDMACM_B/PDMACH_B/INTC2_B/SP*n*_B — P-bus FS Guard Protection Setting Register (*n* = 1, 2, 4)

These registers are setting for P-Bus guard at write cycle.

**Access:** These registers can be Read/written in 32/16/8-bit units.

**Address:** APBFSGDPROT_PDMACM_B: FFC4 C008$_H$
APBFSGDPROT_PDMACH_B: FFC4 C018$_H$
APBFSGDPROT_INTC2_B: FFC4 C028$_H$
APBFSGDPROT_SP1_B: FFC4 C058$_H$
APBFSGDPROT_SP2_B: FFC4 C068$_H$
APBFSGDPROT_SP4_B: FFC4 C078$_H$

**Value after reset:** 060D FE17$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOCK | — | — | — | — | — | — | — | — | PEID5 | PEID4 | — | PEID2 | PEID1 | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.172 APBFSGDPROT_PDMACM_B/PDMACH_B/INTC2_B/SP*n*_B register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | LOCK | Lock Bit<br>0: Registers can be-written<br>1: Any further write to APBFSGDPROT_* and APBFSGDSPID_* are ignored. This bit can only be cleared by reset |
| 30 to 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22, 21 | PEID[5:4] | Access with PEID<br>PEID is a bit list where each bit represents one PEID value. Setting more than one bit allows to enable more than one virtual Processor ID value at a time.<br>0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 20 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 19, 18 | PEID[2:1] | Access with PEID<br>PEID is a bit list where each bit represents one PEID value. Setting more than one bit allows to enable more than one virtual Processor ID value at a time.<br>0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 17 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (3) APBFSGDPROT_PBG_A — P-bus FS Guard Protection Setting Register

This register is setting for P-bus guard at read cycle.

**Access:** This register can be Read/written in 32/16/8-bit units.

**Address:** FFC4 C040$_H$

**Value after reset:** 064D FE1B$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOCK | — | — | — | — | — | — | — | — | PEID5 | PEID4 | — | PEID2 | PEID1 | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.173 APBFSGDPROT_PBG_A register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | LOCK | Lock Bit<br>0: Registers can be-written<br>1: Any further write to APBFSGDPROT_PBG_A and APBFSGDSPID_PBG_A are ignored.<br>This bit can only be cleared by reset |
| 30 to 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22, 21 | PEID[5:4] | Access with PEID<br>PEID is a bit list where each bit represents one PEID value. Setting more than one bit allows to enable more than one virtual Processor ID value at a time.<br>0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 20 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 19, 18 | PEID[2:1] | Access with PEID<br>PEID is a bit list where each bit represents one PEID value. Setting more than one bit allows to enable more than one virtual Processor ID value at a time.<br>0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 17 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (4) APBFSGDPROT_PBG_B — P-bus FS Guard Protection Setting Register

This register is setting for P-bus guard at write cycle.

**Access:** This register can be Read/written in 32/16/8-bit units.

**Address:** FFC4 C048$_H$

**Value after reset:** 064D FE17$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOCK | — | — | — | — | — | — | — | — | PEID5 | PEID4 | — | PEID2 | PEID1 | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.174 APBFSGDPROT_PBG_B register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | LOCK | Lock Bit<br>0: Registers can be-written<br>1: Any further write to APBFSGDPROT_PBG_B and APBFSGDSPID_PBG_B are ignored.<br>This bit can only be cleared by reset |
| 30 to 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22, 21 | PEID[5:4] | Access with PEID<br>PEID is a bit list where each bit represents one PEID value. Setting more than one bit allows to enable more than one virtual Processor ID value at a time.<br>0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 20 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 19, 18 | PEID[2:1] | Access with PEID<br>PEID is a bit list where each bit represents one PEID value. Setting more than one bit allows to enable more than one virtual Processor ID value at a time.<br>0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 17 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (5) APBFSGDSPID_PDMACM_x/PDMACH_x/INTC2_x/PBG_x/SPy_x — P-bus FS Guard SPID Setting Register (x = A or B, y = 1, 2, 4)

"A" is setting for "Read Cycle", "B" is setting for "Write Cycle".

**Access:** These registers can be Read/written in 32/16/8-bit units.

**Address:**

| | |
|---|---|
| APBFSGDSPID_INTC2_A: | FFC4 C024$_H$ |
| APBFSGDSPID_INTC2_B: | FFC4 C02C$_H$ |
| APBFSGDSPID_PDMACM_A: | FFC4 C004$_H$ |
| APBFSGDSPID_PDMACM_B: | FFC4 C00C$_H$ |
| APBFSGDSPID_PDMACH_A: | FFC4 C014$_H$ |
| APBFSGDSPID_PDMACH_B: | FFC4 C01C$_H$ |
| APBFSGDSPID_PBG_A: | FFC4 C044$_H$ |
| APBFSGDSPID_PBG_B: | FFC4 C04C$_H$ |
| APBFSGDSPID_SP1_A: | FFC4 C054$_H$ |
| APBFSGDSPID_SP1_B: | FFC4 C05C$_H$ |
| APBFSGDSPID_SP2_A: | FFC4 C064$_H$ |
| APBFSGDSPID_SP2_B: | FFC4 C06C$_H$ |
| APBFSGDSPID_SP4_A: | FFC4 C074$_H$ |
| APBFSGDSPID_SP4_B: | FFC4 C07C$_H$ |

**Value after reset:** FFFF FFFF$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPID[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.175 APBFSGDSPID_PDMACM_x/PDMACH_x/INTC2_x/PBG_x/SPy_x register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SPID[31:0] | Access with SPID<br>SPID is a bit list where each bit represents one SPID value. Setting more than one bit allows to enable more than one SPID value at a time. E.g. setting SPID to "00000110" allows access with SPID = 1 and SPID = 2.<br>0: Access with SPID n is not allowed.<br>1: Access with SPID n is allowed. |

### (6) ERRSLVCTL_PFSS0/1 — ERRSLV Control Register for PFSS P-bus FS Guard Slave0/1

**Access:** These registers can be written only in 32-bit units.

**Address:** ERRSLVCTL_PFSS0: FFC4 C800$_H$
ERRSLVCTL_PFSS1: FFC4 C810$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLRO | CLRE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W | W |

**Table 28.176 ERRSLVCTL_PFSS0/1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | The write value should also be 0. |
| 1 | CLRO | Clear the OVF bit of ERRSLVSTAT_PFSS0/1 by writing this bit to “1”. |
| 0 | CLRE | Clear the ERR bit of ERRSLVSTAT_PFSS0/1 by writing this bit to “1” |

### (7) ERRSLVSTAT_PFSS0/1 — ERRSLV Status Register for PFSS P-bus FS Guard Slave0/1

**Access:** These registers can be read in 32-bit units.

**Address:** ERRSLVSTAT_PFSS0: FFC4 C804$_H$
ERRSLVSTAT_PFSS1: FFC4 C814$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF | ERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.177 ERRSLVCTL_PFSS0/1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | This bit is always read as 0. |
| 1 | OVF | Error Overflow Status Flag<br>0: No Overflow<br>1: Error Overflow |
| 0 | ERR | Error Status Flag<br>0: No Error<br>1: Error |

### (8) ERRSLVTYPE_PFSS0/1 — ERRSLV Error Transfer Type Register for PFSS P-bus FS Guard Slave0/1

**Access:** These registers can be read in 32-bit units.

**Address:** ERRSLVTYPE_PFSS0: FFC4 C80C$_H$
ERRSLVTYPE_PFSS1: FFC4 C81C$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | SPID[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID[2:0] | | | — | — | — | — | — | — | — | — | — | — | — | — | WRITE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.178 ERRSLVTYPE_PFSS0/1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. |
| 20 to 16 | SPID[4:0] | SPID at P-bus Guard error has occurred |
| 15 to 13 | PEID[2:0] | PEID at P-bus Guard error has occurred |
| 12 to 10 | Reserved | These bits are undefined. |
| 9, 8 | Reserved | When read, the value after reset is read. |
| 7, 6 | Reserved | These bits are undefined. |
| 5 | Reserved | When read, the value after reset is read. |
| 4 to 1 | Reserved | These bits are undefined. |
| 0 | WRITE | Access type at P-bus Guard error<br>0: read access<br>1: write access |

### (9) xxxPROT*x* — P-Bus Guard Protection Setting Register

**Access:** These registers can be Read/written in 32/16/8-bit units.

**Address:** See **Table 28.170, List of Registers** (P-bus Guard)

**Value after reset:** See **Table 28.170, List of Registers** (P-bus Guard)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOCK | — | — | — | — | — | — | — | — | PEID5 | PEID4 | — | PEID2 | PEID1 | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | | | | |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.179 xxxPROTx register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | LOCK | Lock Bit<br>0: Registers can be-written<br>1: Any further write to xxxPROTx and xxxSPIDx are ignored.<br>This bit can only be cleared by reset |
| 30 to 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22, 21 | PEID[5:4] | Access with PEID<br>PEID is a bit list where each bit represents one PEID value. Setting more than one bit allows to enable more than one virtual Processor ID value at a time.<br>0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 20 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 19, 18 | PEID[2:1] | Access with PEID<br>PEID is a bit list where each bit represents one PEID value. Setting more than one bit allows to enable more than one virtual Processor ID value at a time.<br>0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 17 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. See the **Table 28.170** the read values of these bits. |

### (10) xxxSPIDx — P-bus Guard SPID Setting Register

**Access:** These registers can be Read/written in 32-bit units.

**Address:** See **Table 28.170, List of Registers**(P-bus Guard)

**Value after reset:** FFFF FFFF$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPID[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.180 xxxSPIDx register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SPID[31:0] | Access with SPID<br>SPID is a bit list where each bit represents one SPID value. Setting more than one bit allows to enable more than one SPID value at a time. E.g. setting SPID to “00000110” allows access with SPID = 1 and SPID = 2.<br>0: Access with SPID n is not allowed.<br>1: Access with SPID n is allowed. |

### (11) ERRSLVxCTL — ERRSLV Control Register for P-bus Guard

**Access:** These registers can be written only in 8-bit units.

**Address:** See **Table 28.170, List of Registers** (P-bus Guard)

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CLRO | CLRE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | W | W |

**Table 28.181 ERRSLVxCTL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | The write value should also be 0. |
| 1 | CLRO | Clear the OVF bit of ERRSLVxSTAT by writing this bit to “1”. |
| 0 | CLRE | Clear the ERR bit of ERRSLVxSTAT by writing this bit to “1” |

### (12) ERRSLVxSTAT — ERRSLV Status Register for P-Bus Guard

**Access:** These registers can be read in 32-bit units.

**Address:** See **Table 28.170, List of Registers** (P-bus Guard)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF | ERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.182 ERRSLV1STAT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | This bit is always read as 0. |
| 1 | OVF | Error Overflow Status Flag<br>0: No Overflow<br>1: Error Overflow |
| 0 | ERR | Error Status Flag<br>0: No Error<br>1: Error |

### (13) ERRSLVxTYPE — ERRSLV Error Transfer Type Register for P-Bus Guard

**Access:** These registers can be read in 32-bit units.

**Address:** See **Table 28.170, List of Registers** (P-bus Guard)

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | SPID[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID[2:0] | | | — | — | — | — | — | — | — | — | — | — | — | — | WRITE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.183 ERRSLVxTYPE register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. |
| 20 to 16 | SPID[4:0] | SPID at P-bus Guard error has occurred |
| 15 to 13 | PEID[2:0] | PEID at P-bus Guard error has occurred |
| 12 to 10 | Reserved | Reserved. These bits are undefined. |
| 9, 8 | Reserved | When read, the value after reset is read. |
| 7, 6 | Reserved | Reserved. These bits are undefined. |
| 5 | Reserved | When read, the value after reset is read. |
| 4 to 1 | Reserved | Reserved. These bits are undefined. |
| 0 | WRITE | Access type at P-bus Guard error<br>0: read access<br>1: write access |

### 28.4.4 HBG

HBUS guard can protect respectively the read and write access against which a peripheral circuit on HBUS should be protected. If HBUS guard detects illegal access, guard error notification is signaled to ECM.

**Table 28.184 List of Registers (1/3)**

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | HBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFFA 0000$_H$ | FSGDF0PROT00 | Flexray0 H-bus Guard for Read | R/W | 060D FE1B$_H$ | 8/16/32 | FlexRay0 | **28.4.4.1 (1)** | |
| FFFA 0004$_H$ | FSGDF0SPID00 | Flexray0 H-bus Guard for Read | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | |
| FFFA 0008$_H$ | FSGDF0PROT01 | Flexray0 H-bus Guard for Write | R/W | 060D FE17$_H$ | 8/16/32 | | **28.4.4.1 (2)** | |
| FFFA 000C$_H$ | FSGDF0SPID01 | Flexray0 H-bus Guard for Write | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | |
| FFFA 0010$_H$ | ERRSLVF0CTL | Flexray0 H-bus Guard Error slave | W | 00$_H$ | 8 | | **28.4.4.1 (4)** | |
| FFFA 0014$_H$ | ERRSLVF0STAT | Flexray0 H-bus Guard Error slave | R | 0000 0000$_H$ | 32 | | **28.4.4.1 (5)** | |
| FFFA 001C$_H$ | ERRSLVF0TYPE | Flexray0 H-bus Guard Error slave | R | 0000 0000$_H$ | 32 | | **28.4.4.1 (6)** | |
| FFFA 0020$_H$ | FSGDF1PROT00 | Flexray1 H-bus Guard for Read | R/W | 060D FE1B$_H$ | 8/16/32 | FlexRay1 | **28.4.4.1 (1)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0024$_H$ | FSGDF1SPID00 | Flexray1 H-bus Guard for Read | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0028$_H$ | FSGDF1PROT01 | Flexray1 H-bus Guard for Write | R/W | 060D FE17$_H$ | 8/16/32 | | **28.4.4.1 (2)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 002C$_H$ | FSGDF1SPID01 | Flexray1 H-bus Guard for Write | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0030$_H$ | ERRSLVF1CTL | Flexray1 H-bus Guard Error slave | W | 00$_H$ | 8 | | **28.4.4.1 (4)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0034$_H$ | ERRSLVF1STAT | Flexray1 H-bus Guard Error slave | R | 0000 0000$_H$ | 32 | | **28.4.4.1 (5)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 003C$_H$ | ERRSLVF1TYPE | Flexray1 H-bus Guard Error slave | R | 0000 0000$_H$ | 32 | | **28.4.4.1 (6)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0040$_H$ | FSGDE0PROT00 | Ethernet0 H-bus Guard for Read | R/W | 060D FE1B$_H$ | 8/16/32 | Ethernet0 | **28.4.4.1 (1)** | |
| FFFA 0044$_H$ | FSGDE0SPID00 | Ethernet0 H-bus Guard for Read | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | |
| FFFA 0048$_H$ | FSGDE0PROT01 | Ethernet0 H-bus Guard for Write | R/W | 060D FE17$_H$ | 8/16/32 | | **28.4.4.1 (2)** | |
| FFFA 004C$_H$ | FSGDE0SPID01 | Ethernet0 H-bus Guard for Write | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | |
| FFFA 0050$_H$ | ERRSLVE0CTL | Ethernet0 H-bus Guard Error slave | W | 00$_H$ | 8 | | **28.4.4.1 (4)** | |
| FFFA 0054$_H$ | ERRSLVE0STAT | Ethernet0 H-bus Guard Error slave | R | 0000 0000$_H$ | 32 | | **28.4.4.1 (5)** | |
| FFFA 005C$_H$ | ERRSLVE0TYPE | Ethernet0 H-bus Guard Error slave | R | 0000 0000$_H$ | 32 | | **28.4.4.1 (6)** | |

Table 28.184 List of Registers (2/3)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | HBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFFA 0060$_H$ | FSGDE1PROT00 | Ethernet1 H-bus Guard for Read | R/W | 060D FE1B$_H$ | 8/16/32 | Ethernet1 | **28.4.4.1 (1)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0064$_H$ | FSGDE1SPID00 | Ethernet1 H-bus Guard for Read | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0068$_H$ | FSGDE1PROT01 | Ethernet1 H-bus Guard for Write | R/W | 060D FE17$_H$ | 8/16/32 | | **28.4.4.1 (2)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 006C$_H$ | FSGDE1SPID01 | Ethernet1 H-bus Guard for Write | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0070$_H$ | ERRSLVE1CTL | Ethernet1 H-bus Guard Error slave | W | 00$_H$ | 8 | | **28.4.4.1 (4)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0074$_H$ | ERRSLVE1STAT | Ethernet1 H-bus Guard Error slave | R | 0000 0000$_H$ | 32 | | **28.4.4.1 (5)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 007C$_H$ | ERRSLVE1TYPE | Ethernet1 H-bus Guard Error slave | R | 0000 0000$_H$ | 32 | | **28.4.4.1 (6)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0080$_H$ | FSGDM0PROT00 | MEMC H-bus Guard for CS0 for Read | R/W | 060D FE1B$_H$ | 8/16/32 | MEMC | **28.4.4.1 (1)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0084$_H$ | FSGDM0SPID00 | MEMC H-bus Guard for CS0 for Read | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0088$_H$ | FSGDM0PROT01 | MEMC H-bus Guard for CS0 for Write | R/W | 060D FE17$_H$ | 8/16/32 | | **28.4.4.1 (2)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 008C$_H$ | FSGDM0SPID01 | MEMC H-bus Guard for CS0 for Write | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0090$_H$ | FSGDM0PROT02 | MEMC H-bus Guard for CS1 for Read | R/W | 060D FE1B$_H$ | 8/16/32 | | **28.4.4.1 (1)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0094$_H$ | FSGDM0SPID02 | MEMC H-bus Guard for CS1 for Read | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 0098$_H$ | FSGDM0PROT03 | MEMC H-bus Guard for CS1 for Write | R/W | 060D FE17$_H$ | 8/16/32 | | **28.4.4.1 (2)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 009C$_H$ | FSGDM0SPID03 | MEMC H-bus Guard for CS1 for Write | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00A0$_H$ | FSGDM0PROT04 | MEMC H-bus Guard for CS2 for Read | R/W | 060D FE1B$_H$ | 8/16/32 | | **28.4.4.1 (1)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00A4$_H$ | FSGDM0SPID04 | MEMC H-bus Guard for CS2 for Read | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00A8$_H$ | FSGDM0PROT05 | MEMC H-bus Guard for CS2 for Write | R/W | 060D FE17$_H$ | 8/16/32 | | **28.4.4.1 (2)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00AC$_H$ | FSGDM0SPID05 | MEMC H-bus Guard for CS2 for Write | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00B0$_H$ | FSGDM0PROT06 | MEMC H-bus Guard for CS3 for Read | R/W | 060D FE1B$_H$ | 8/16/32 | | **28.4.4.1 (1)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00B4$_H$ | FSGDM0SPID06 | MEMC H-bus Guard for CS3 for Read | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |

Table 28.184 List of Registers (3/3)

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | HBG | Refer to Register | Note |
|---|---|---|---|---|---|---|---|---|
| FFFA 00B8$_H$ | FSGDM0PROT07 | MEMC H-bus Guard for CS3 for Write | R/W | 060D FE17$_H$ | 8/16/32 | MEMC | **28.4.4.1 (2)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00BC$_H$ | FSGDM0SPID07 | MEMC H-bus Guard for CS3 for Write | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00C0$_H$ | FSGDM0PROT08 | MEMC H-bus Guard for REG for Read | R/W | 060D FE1B$_H$ | 8/16/32 | | **28.4.4.1 (1)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00C4$_H$ | FSGDM0SPID08 | MEMC H-bus Guard for REG for Read | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00C8$_H$ | FSGDM0PROT09 | MEMC H-bus Guard for REG for Write | R/W | 060D FE17$_H$ | 8/16/32 | | **28.4.4.1 (2)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00CC$_H$ | FSGDM0SPID09 | MEMC H-bus Guard for REG for Write | R/W | FFFF FFFF$_H$ | 8/16/32 | | **28.4.4.1 (3)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00D0$_H$ | ERRSLVM0CTL | MEMC H-bus Guard Error slave | W | 00$_H$ | 8 | | **28.4.4.1 (4)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00D4$_H$ | ERRSLVM0STAT | MEMC H-bus Guard Error slave | R | 0000 0000$_H$ | 32 | | **28.4.4.1 (5)** | except P1M-C, P1H-C (4MB, BGA-156) |
| FFFA 00DC$_H$ | ERRSLVM0TYPE | MEMC H-bus Guard Error slave | R | 0000 0000$_H$ | 32 | | **28.4.4.1 (6)** | except P1M-C, P1H-C (4MB, BGA-156) |

### 28.4.4.1 Details of Registers

#### (1) FSGDxxPROTn (n = 00, 02, 04, 06, 08)

**Access:** These registers can be Read/written in 32/16/8-bit units.

**Address:** See Table 28.183, HBus Guard.

**Value after reset:** 060D FE1B$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOCK | — | — | — | — | — | — | — | — | PEID[5:4] | | — | PEID[2:1] | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.185 FSGDxxPROTnn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | LOCK | Lock Bit<br>0: Registers can be-written<br>1: Any further write to FSGDxxPROTn and FSGDxxSPIDn are ignored. This bit can only be cleared by reset |
| 30 to 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22, 21 | PEID[5:4] | 0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 20 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 19, 18 | PEID[2:1] | 0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 17 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (2) FSGDxxPROTn (n = 01, 03, 05, 07, 09)

**Access:** These registers can be Read/written in 32/16/8-bit units.

**Address:** See Table 28.183, HBus Guard.

**Value after reset:** 060D FE17$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOCK | — | — | — | — | — | — | — | — | PEID[5:4] | | — | PEID[2:1] | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.186 FSGDxxPROTnn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | LOCK | Lock Bit<br>0: Registers can be-written<br>1: Any further write to FSGDxxPROTn and FSGDxxSPIDn are ignored. This bit can only be cleared by reset |
| 30 to 23 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 22, 21 | PEID[5:4] | 0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 20 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 19, 18 | PEID[2:1] | 0: Access with PEID n is not allowed.<br>1: Access with PEID n is allowed. |
| 17 to 0 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |

### (3) FSGDxxSPIDnn

**Access:** These registers can be Read/written in 32/16/8-bit units.

**Address:** See Table 28.183, HBus Guard.

**Value after reset:** FFFF FFFF$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPID[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 28.187 FSGDxxSPIDnn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | SPID[31:0] | Access with SPID<br>SPID is a bit list where each bit represents one SPID value. Setting more than one bit allows to enable more than one SPID value at a time. E.g. setting SPID to “00000110” allows access with SPID = 1 and SPID = 2.<br>0: Access with SPID n is not allowed.<br>1: Access with SPID n is allowed. |

### (4) ERRSLVxxCTL

**Access:** These registers can be written in 8-bit units.

**Address:** See Table 28.183, HBus Guard.

**Value after reset:** 00$_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CLRO | CLRE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | W | W |

**Table 28.188 ERRSLVxxCTL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 2 | Reserved | The write value should also be 0. |
| 1 | CLRO | 0: Ignored<br>1: Clears the OVF bit of ERRSLVxxSTAT<br>Always read “0” |
| 0 | CLRE | 0: Ignored<br>1: Clears the ERR bit of ERRSLVxxSTAT<br>Always read “0” |

### (5) ERRSLVxxSTAT

**Access:** These registers can be read in 32-bit units.

**Address:** See Table 28.183, HBus Guard.

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF | ERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.189 ERRSLVxxSTAT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | This bit is always read as 0. |
| 1 | OVF | Error Overflow Status Flag<br>0: No Overflow<br>1: Error Overflow |
| 0 | ERR | Error Status Flag<br>0: No Error<br>1: Error |

### (6) ERRSLVxxTYPE

**Access:** These registers can be read in 32-bit units.

**Address:** See Table 28.183, HBus Guard.

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | SPID[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEID[2:0] | | | — | — | — | — | — | — | — | — | — | — | — | — | WRITE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.190 ERRSLVxxTYPE register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 21 | Reserved | When read, the value after reset is read. |
| 20 to 16 | SPID[4:0] | SPID at H-Bus Guard error has occurred |
| 15 to 13 | PEID[2:0] | PEID at H-Bus Guard error has occurred |
| 12 to 10 | Reserved | These bits are undefined. |
| 9, 8 | Reserved | When read, the value after reset is read. |
| 7, 6 | Reserved | These bits are undefined. |
| 5, 4 | Reserved | When read, the value after reset is read. |
| 3 to 1 | Reserved | These bits are undefined. |
| 0 | WRITE | Access type at H-bus Guard error<br>0: read access<br>1: write access |

### (7) HSSPIDRGn (n = 0 to 7)

This register is setting SPID for each HBUS master IP(Flexray0/1,Ethernet0/1,HS-USRT0-3).The HBUS master IPs can only use a value of SPID between 2 and 31.

**Access:** These registers can be Read/written in 32-bit units.

**Address:** HSSPIDRG0: FFFA $1000_H$
HSSPIDRG1: FFFA $1004_H$
HSSPIDRG2: FFFA $1008_H$
HSSPIDRG3: FFFA $100C_H$
HSSPIDRG4: FFFA $1010_H$
HSSPIDRG5: FFFA $1014_H$
HSSPIDRG6: FFFA $1018_H$
HSSPIDRG7: FFFA $101C_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | SPID[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 28.191 HSSPIDRGn register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 4 to 0 | SPID[4:0] | System Protection ID.<br>In case that this value is “0” or “1”, SPID of the master is “2” instead of register value. |

**Table 28.192 Target IP**

| Register Symbol | target IP |
|---|---|
| HSSPIDRG0 | FlexRay0 |
| HSSPIDRG1 | FlexRay1 |
| HSSPIDRG2 | Ethernet0 |
| HSSPIDRG3 | Ethernet1 |
| HSSPIDRG4 | HS-USRT0 |
| HSSPIDRG5 | HS-USRT1 |
| HSSPIDRG6 | HS-USRT2 |
| HSSPIDRG7 | HS-USRT3 |

## 28.5 BIST

This product incorporates the function to detect failures of the failure detection function itself, which is referred to as BIST. The Memory BIST is implemented for all RAMs except Emulation RAM and Trace RAM.

BIST execution results can be identified by the field BIST result register (BSEQ0ST).

When the internal oscillator is selected as clock source of system clocks (CKSC0S = $2_H$), Field BIST fails because BIST sequence does not finished in the range of time.

### 28.5.1 List of Registers

Table 28.193 List of Registers

| Address | Symbol | Register Name | R/W | Value after reset | Access Size | PBG | Other Protection | Note |
|---|---|---|---|---|---|---|---|---|
| FFCD A000$_H$ | LBISTREF1 | Logic BIST reference value register 1 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A004$_H$ | LBISTREF2 | Logic BIST reference value register 2 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A008$_H$ | MBISTREF1 | Memory BIST reference value register 1 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A00C$_H$ | MBISTREF2 | Memory BIST reference value register 2 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A010$_H$ | LBISTSIG1 | Logic BIST signature value register 1 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A014$_H$ | LBISTSIG2 | Logic BIST signature value register 2 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A018$_H$ | MBISTSIG1 | Memory BIST signature value register 1 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A01C$_H$ | MBISTSIG2 | Memory BIST signature value register 2 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A020$_H$ | MBISTFTAGL1 | Memory BIST FTAG signature value register L 1 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A024$_H$ | MBISTFTAGH1 | Memory BIST FTAG signature value register H 1 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A028$_H$ | MBISTFTAGL2 | Memory BIST FTAG signature value register L 2 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A02C$_H$ | MBISTFTAGH2 | Memory BIST FTAG signature value register H 2 | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A030$_H$ | BSEQ0ST | BIST Sequence status register | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A034$_H$ | BSEQ0STB | BIST Sequence inverted status register | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFCD A038$_H$ | BISTST | Field BIST Result register | R | *1 | 32 | PBG4#0.PG4-Startup | | |
| FFF8 0200$_H$ | BSEQ0CTL | Field BIST control register | R/W | 0000 0001$_H$ | 32 | PBG4#0.PG4-SC3 | | |

Note 1. This register doesn't show the fixed value.

## 28.5.2 Details of Registers

### (1) LBISTREF1 — Logic BIST reference value register 1

This register indicates the reference value of the Logic BIST.

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A000$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | LBISTREF1[19:16] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LBISTREF1[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.194 LBISTREF1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset is read. |
| 19 to 0 | LBISTREF1 [19:0] | Reference signature (expected signature) |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2]
[Debug mode(TRSTZ = 1)]
It is not matched with the value of corresponding signature register.
[Application reset1]
The previous value is kept.

### (2) LBISTREF2 — Logic BIST reference value register 2

This register indicates the reference value of the Logic BIST.

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A004$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | LBISTREF2[19:16] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LBISTREF2[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.195 LBISTREF2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset is read. |
| 19 to 0 | LBISTREF2 [19:0] | Reference signature (expected signature) |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2]
[Debug mode(TRSTZ = 1)]
It is not matched with the value of corresponding signature register.
[Application reset1]
The previous value is kept.

### (3) MBISTREF1 — Memory BIST reference value register 1

This register indicates the reference value of the Memory BIST.

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A008$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MBISTREF1[19:16] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTREF1[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.196 MBISTREF1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset is read. |
| 19 to 0 | MBISTREF1 [19:0] | Reference signature (expected signature) |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2]
[Debug mode (TRSTZ = 1)]
It is not matched with the value of corresponding signature register.
[Application reset1]
The previous value is kept.

### (4) MBISTREF2 — Memory BIST reference value register 2

This register indicates the reference value of the Memory BIST.

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A00C$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MBISTREF2[19:16] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTREF2[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.197 MBISTREF2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset is read. |
| 19 to 0 | MBISTREF2 [19:0] | Reference signature (expected signature) |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2]
[Debug mode (TRSTZ = 1)]
It is not matched with the value of corresponding signature register.
[Application reset1]
The previous value is kept.

### (5) LBISTSIG1 — Logic BIST signature value register 1

This register indicates the signature value of the Logic BIST. The user shall compare the reference signature of LBISTREF1 against the resulting signature LBISTSIG1. The LBIST is passed if these are equal.

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A010$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | LBISTSIG1[19:16] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LBISTSIG1[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.198 LBISTSIG1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset is read. |
| 19 to 0 | LBISTSIG1 [19:0] | LBIST1 signature result |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2]
[Debug mode (TRSTZ = 1)]
It is not matched with the value of corresponding reference register.
[Application reset1]
The previous value is kept.

### (6) LBISTSIG2 — Logic BIST signature value register 2

This register indicates the signature value of the Logic BIST. The user shall compare the reference signature of LBISTREF2 against the resulting signature LBISTSIG2. The LBIST is passed if these are equal.

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A014$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | LBISTSIG2[19:16] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LBISTSIG2[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.199 LBISTSIG2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset is read. |
| 19 to 0 | LBISTSIG2 [19:0] | LBIST2 signature result |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2]
[Debug mode(TRSTZ = 1)]
It is not matched with the value of corresponding reference register.
[Application reset1]
The previous value is kept.

### (7) MBISTSIG1 — Memory BIST signature value register 1

This register indicates the signature value of the Memory BIST. The user shall compare the reference signature of MBISTREF1 against the resulting signature MBISTSIG1. The MBIST is passed if these are equal.

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A018$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MBISTSIG1[19:16] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTSIG1[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.200 MBISTSIG1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset is read. |
| 19 to 0 | MBISTSIG1 [19:0] | MBIST1 signature result |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2]
[Debug mode(TRSTZ = 1)]
It is not matched with the value of corresponding reference register.
[Application reset1]
The previous value is kept.

### (8) MBISTSIG2 — Memory BIST signature value register 2

This register indicates the signature value of the Memory BIST. The user shall compare the reference signature of MBISTREF2 against the resulting signature MBISTSIG2. The MBIST is passed if these are equal.

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A01C$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MBISTSIG2[19:16] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTSIG2[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.201 MBISTSIG2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | When read, the value after reset is read. |
| 19 to 0 | MBISTSIG2 [19:0] | MBIST2 signature result |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2]
[Debug mode(TRSTZ = 1)]
It is not matched with the value of corresponding reference register.
[Application reset1]
The previous value is kept.

### (9) MBISTFTAGL1 — Memory BIST FTAG signature value register L1

This register indicates the Memory BIST status of each RAM group (bridge).

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A020$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTFTAGL1[31:16] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTFTAGL1[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.202 MBISTFTAGL1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | MBISTFTAGL1 [31:0] | The state of self-diagnostic MBIST of Bridge No.n<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL.<br>Refer to **Table 28.203** for mapping of each RAM. |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2] or [Debug mode (TRSTZ = 1)]
This register shows FFFF FFFF$_H$.
[Application reset1]
The previous value is kept.

**Table 28.203 mapping of MBISTFTAGL1 register (1/3)**

| Bit Position | Bit Name | Function | | | |
|---|---|---|---|---|---|
| | | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
| 31 | MBISTFTAGL1 [31] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEF3_0000$_H$ to FEF3_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 30 | MBISTFTAGL1 [30] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEEB_0000$_H$ to FEEB_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 29 | MBISTFTAGL1 [29] | The state of self-diagnostic MBIST of Global RAM which address is from FEEF_0000$_H$ to FEEF_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 28 | MBISTFTAGL1 [28] | The state of self-diagnostic MBIST of ICUMC Instruction cache RAM (data).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 27 | MBISTFTAGL1 [27] | The state of self-diagnostic MBIST of ICUMC Instruction cache RAM (tag).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 26 | MBISTFTAGL1 [26] | The state of self-diagnostic MBIST of ICUMC Instruction cache RAM (data).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 25 | MBISTFTAGL1 [25] | The state of self-diagnostic MBIST of ICUMC Instruction cache RAM (data).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |

Table 28.203 mapping of MBISTFTAGL1 register (2/3)

| Bit Position | Bit Name | Function | | | |
|---|---|---|---|---|---|
| | | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
| 24 | MBISTFTAGL1 [24] | The state of self-diagnostic MBIST of MCAN RAM (MTTCAN0, MCAN1).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | The state of self-diagnostic MBIST of MCAN RAM (MTTCAN0).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 23 | MBISTFTAGL1 [23] | The state of self-diagnostic MBIST of GTM RAM.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 22 | MBISTFTAGL1 [22] | The state of self-diagnostic MBIST of Ethernet RAM (Retry RAM for port0).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | The state of self-diagnostic MBIST of Ethernet RAM (Retry RAM for port0 and port1).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 21 | MBISTFTAGL1 [21] | The state of self-diagnostic MBIST of Ethernet RAM (Receive/Transmit FIFO for port0).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | The state of self-diagnostic MBIST of Ethernet RAM (Receive/Transmit FIFO for port0 and port1).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 20 | MBISTFTAGL1 [20] | The state of self-diagnostic MBIST of CSIH RAM.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 19 | MBISTFTAGL1 [19] | The state of self-diagnostic MBIST of FlexRay RAM (Temporary buffer RAM for FLX0).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | The state of self-diagnostic MBIST of FlexRay RAM (Temporary buffer RAM for FLX0 and FLX1).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 18 | MBISTFTAGL1 [18] | The state of self-diagnostic MBIST of FlexRay RAM(Message RAM for FLX0) or MCAN RAM(MCAN0).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | The state of self-diagnostic MBIST of FlexRay RAM (Message RAM for FLX0 and FLX1) or MCAN RAM(MCAN0/1).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 17 | MBISTFTAGL1 [17] | The state of self-diagnostic MBIST of Local RAM (ICUMC).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 16 | MBISTFTAGL1 [16] | The state of self-diagnostic MBIST of DTS RAM.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 15 | MBISTFTAGL1 [15] | The state of self-diagnostic MBIST of FCU RAM.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 14 | MBISTFTAGL1 [14] | Reserved Bit*1 | Reserved Bit*1 | Reserved Bit*1 | Reserved Bit*1 |
| 13 | MBISTFTAGL1 [13] | Reserved Bit*1 | Reserved Bit*1 | Reserved Bit*1 | Reserved Bit*1 |
| 12 | MBISTFTAGL1 [12] | Reserved Bit*1 | Reserved Bit*1 | Reserved Bit*1 | Reserved Bit*1 |
| 11 | MBISTFTAGL1 [11] | Reserved Bit*1 | The state of self-diagnostic MBIST of Local RAM (PE2) which 4bit of address LSB is $0_H$ or $4_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 10 | MBISTFTAGL1 [10] | Reserved Bit*1 | The state of self-diagnostic MBIST of Local RAM (PE2) which 4bit of address LSB is $8_H$ or $C_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 9 | MBISTFTAGL1 [9] | Reserved Bit*1 | The state of self-diagnostic MBIST of Instruction cache RAM (tag) (PE2).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 8 | MBISTFTAGL1 [8] | The state of self-diagnostic MBIST of Instruction cache RAM (data) (PE1).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 7 | MBISTFTAGL1 [7] | The state of self-diagnostic MBIST of Instruction cache RAM (data) (PE1).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |

**Table 28.203 mapping of MBISTFTAGL1 register (3/3)**

| Bit Position | Bit Name | Function | | | |
|---|---|---|---|---|---|
| | | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
| 6 | MBISTFTAGL1 [6] | Reserved Bit*1 | The state of self-diagnostic MBIST of Instruction cache RAM (data) (PE2).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 5 | MBISTFTAGL1 [5] | Reserved Bit*1 | The state of self-diagnostic MBIST of Instruction cache RAM (data) (PE2).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 4 | MBISTFTAGL1 [4] | The state of self-diagnostic MBIST of Local RAM (PE1) which address is from FEBF_0000$_H$ to FEBF_FFFC$_H$ and 4bit of address LSB is 0$_H$ or 4$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 3 | MBISTFTAGL1 [3] | The state of self-diagnostic MBIST of Local RAM (PE1) which address is from FEBF_0000$_H$ to FEBF_FFFC$_H$ and 4bit of address LSB is 8$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 2 | MBISTFTAGL1 [2] | The state of self-diagnostic MBIST of Instruction cache RAM(tag)(PE1).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 1 | MBISTFTAGL1 [1] | Reserved Bit*1 | Reserved Bit*1 | Reserved Bit*1 | Reserved Bit*1 |
| 0 | MBISTFTAGL1 [0] | Reserved Bit*1 | Reserved Bit*1 | The state of self-diagnostic MBIST of MCAN RAM (MCAN2).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | |

Note 1. When Field BIST is executed, the value is 0.

### (10) MBISTFTAGH1 — Memory BIST FTAG signature value register H1

This register indicates the Memory BIST status of each RAM group (bridge).

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A024$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTFTAGH1[31:16] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTFTAGH1[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.204 MBISTFTAGH1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | MBISTFTAGH1 [31:0] | The state of self-diagnostic MBIST of Bridge No.n<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL.<br>Refer to **Table 28.205** for mapping of each RAM. |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2] or [Debug mode(TRSTZ = 1)]
This register shows FFFF FFFF$_H$.
[Application reset1]
The previous value is kept.

**Table 28.205 mapping of MBISTFTAGH1 register (1/3)**

| Bit Position | Bit Name | Function | | | |
|---|---|---|---|---|---|
| | | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
| 31, 30 | MBISTFTAGH1 [31:30] | The state of self-diagnostic MBIST of Local RAM (PE1) which address is from FEBE_0000$_H$ to FEBE_FFFC$_H$.<br>00: MBIST status is PASS.<br>01,10,11: MBIST status is FAIL. | Reserved Bit*1 | Reserved Bit*1 | The state of self-diagnostic MBIST of Local RAM (PE1) which address is from FEBE_0000$_H$ to FEBE_FFFC$_H$.<br>00: MBIST status is PASS.<br>01,10,11: MBIST status is FAIL. |
| 29 | MBISTFTAGH1 [29] | The state of self-diagnostic MBIST of Local RAM (ICUMC).<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 28 | MBISTFTAGH1 [28] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEF3_0000$_H$ to FEF3_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 27 | MBISTFTAGH1 [27] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEEC_0000$_H$ to FEEC_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 26 | MBISTFTAGH1 [26] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEF7_0000$_H$ to FEF7_7FFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |

Table 28.205 mapping of MBISTFTAGH1 register (2/3)

| Bit Position | Bit Name | Function | | | |
|---|---|---|---|---|---|
| | | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
| 25 | MBISTFTAGH1 [25] | The state of self-diagnostic MBIST of Global RAM which address is from FEF2_0000$_H$ to FEF2_7FFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | The state of self-diagnostic MBIST of Global RAM which address is from FEF2_0000$_H$ to FEF2_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 24 | MBISTFTAGH1 [24] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEEC_0000$_H$ to FEEC_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 23 | MBISTFTAGH1 [23] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEF7_0000$_H$ to FEF7_7FFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 22 | MBISTFTAGH1 [22] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEF6_0000$_H$ to FEF6_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 21 | MBISTFTAGH1 [21] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEE8_8000H to FEE8_FFFCH and is 4bit of address LSB is 0H or 8H.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 20 | MBISTFTAGH1 [20] | The state of self-diagnostic MBIST of Global RAM which address is from FEF1_0000$_H$ to FEF1_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 19 | MBISTFTAGH1 [19] | The state of self-diagnostic MBIST of Global RAM which address is from FEF2_0000$_H$ to FEF2_7FFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | The state of self-diagnostic MBIST of Global RAM which address is from FEF2_0000$_H$ to FEF2_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 18 | MBISTFTAGH1 [18] | The state of self-diagnostic MBIST of Global RAM which address is from FEED_8000$_H$ to FEED_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | The state of self-diagnostic MBIST of Global RAM which address is from FEED_0000$_H$ to FEED_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 17 | MBISTFTAGH1 [17] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEE8_8000$_H$ to FEE8_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 16 | MBISTFTAGH1 [16] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEF6_0000$_H$ to FEF6_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 15 | MBISTFTAGH1 [15] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEE9_0000$_H$ to FEE9_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 14 | MBISTFTAGH1 [14] | The state of self-diagnostic MBIST of Global RAM which address is from FEF1_0000$_H$ to FEF1_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 13 | MBISTFTAGH1 [13] | The state of self-diagnostic MBIST of Global RAM which address is from FEED_8000$_H$ to FEED_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | The state of self-diagnostic MBIST of Global RAM which address is from FEED_0000$_H$ to FEED_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |

**Table 28.205 mapping of MBISTFTAGH1 register (3/3)**

| Bit Position | Bit Name | Function | | | |
|---|---|---|---|---|---|
| | | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
| 12 | MBISTFTAGH1 [12] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEF5_0000$_H$ to FEF5_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 11 | MBISTFTAGH1 [11] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEE9_0000$_H$ to FEE9_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 10 | MBISTFTAGH1 [10] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEEA_0000$_H$ to FEEA_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 9 | MBISTFTAGH1 [9] | The state of self-diagnostic MBIST of Global RAM which address is from FEF0_0000$_H$ to FEF0_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 8 | MBISTFTAGH1 [8] | The state of self-diagnostic MBIST of Global RAM which address is from FEEE_0000$_H$ to FEEE_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 7 | MBISTFTAGH1 [7] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEF5_0000$_H$ to FEF5_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 6 | MBISTFTAGH1 [6] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEEA_0000$_H$ to FEEA_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 5 | MBISTFTAGH1 [5] | The state of self-diagnostic MBIST of Global RAM which address is from FEF0_0000$_H$ to FEF0_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 4 | MBISTFTAGH1 [4] | The state of self-diagnostic MBIST of Global RAM which address is from FEEE_0000$_H$ to FEEE_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 3 | MBISTFTAGH1 [3] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEF4_0000$_H$ to FEF4_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 2 | MBISTFTAGH1 [2] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEEB_0000$_H$ to FEEB_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |
| 1 | MBISTFTAGH1 [1] | The state of self-diagnostic MBIST of Global RAM which address is from FEEF_0000$_H$ to FEEF_FFFC$_H$ and is 4bit of address LSB is 4$_H$ or C$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | | |
| 0 | MBISTFTAGH1 [0] | Reserved Bit*1 | The state of self-diagnostic MBIST of Global RAM which address is from FEF4_0000$_H$ to FEF4_FFFC$_H$ and is 4bit of address LSB is 0$_H$ or 8$_H$.<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL. | | |

Note 1. When Field BIST is executed, the value is 0.

### (11) MBISTFTAGL2 — Memory BIST FTAG signature value register L2

This register indicates the Memory BIST status of each RAM group (bridge).

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A028$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTFTAGL2[31:16] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTFTAGL2[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.206 MBISTFTAGL2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | MBISTFTAGL2 [31:0] | The state of self-diagnostic MBIST of Bridge No.n<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL.<br>Refer to **Table 28.207** for mapping of each RAM. |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2] or [Debug mode (TRSTZ = 1)]
This register shows FFFF FFFF$_H$.
[Application reset1]
The previous value is kept.

**Table 28.207 mapping of MBISTFTAGL2 register**

| Bit Position | Bit Name | Function | | |
|---|---|---|---|---|
| | | P1M-C | P1H-C (4MB) | P1H-C (8MB) |
| 31:0 | MBISTFTAGL2 [31:0] | Reserved Bit*1 | | |

Note 1. When Field BIST is executed, the value is 0.

### (12) MBISTFTAGH2 — Memory BIST FTAG signature value register H2

This register indicates the Memory BIST status of each RAM group (bridge).

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A02C$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTFTAGH2[31:16] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MBISTFTAGH2[15:0] | | | | | | | | | | | | | | | |
| Value after reset | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.208 MBISTFTAGH2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | MBISTFTAGH2 [31:0] | The state of self-diagnostic MBIST of Bridge No.n<br>0: MBIST status is PASS.<br>1: MBIST status is FAIL.<br>Refer to **Table 28.209** for mapping of each RAM. |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2] or [Debug mode (TRSTZ = 1)]
This register shows FFFF FFFF$_H$.
[Application reset1]
The previous value is kept.

**Table 28.209 mapping of MBISTFTAGH2 register**

| Bit Position | Bit Name | Function | | |
|---|---|---|---|---|
| | | P1M-C | P1H-C (4MB) | P1H-C (8MB) |
| 31:0 | MBISTFTAGH2 [31:0] | Reserved Bit*1 | | |

Note 1. When Field BIST is executed, the value is 0.

### (13) BSEQ0ST — BIST Sequencer Status Register

This register indicates the state of the BIST sequencer

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A030$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | BISTEND | CMPERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.210 BSEQ0ST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. |
| 1 | BISTEND | 0: BIST sequence has not finished in the rage of time<br>1: BIST sequence has finished in the rage of time |
| 0 | CMPERR | 0: BIST normal end<br>1: BIST abnormal end. |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2]
This register shows 0000 0001$_H$.
[Debug mode(TRSTZ = 1)]
This register shows 0000 0002$_H$.
[Application reset1]
The previous value is kept.

### (14) BSEQ0STB — BIST Sequencer Inverted Status Register

This register indicates the inverted state of the BIST sequencer

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A034$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | BISTENDB | CMPERRB |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.211 BSEQ0STB register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | When read, the value after reset is read. |
| 1 | BISTENDB | 0: BIST sequence has finished in the range of time<br>1: BIST sequence has not finished in the range of time |
| 0 | CMPERRB | 0: BIST abnormal end<br>1: BIST normal end |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2]
This register shows 0000 0002$_H$.
[Debug mode(TRSTZ = 1)]
This register shows 0000 0001$_H$.
[Application reset1]
The previous value is kept.

### (15) BISTST— Field BIST result register

This register indicates the result of the field BIST.

**Access:** This register can be read only in 32-bit units.

**Address:** FFCD A038$_H$

**Value after reset:** *1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MBIST2ST | MBIST1ST | LBIST2ST | LBIST1ST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | *1 | *1 | *1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 28.212 BISTST register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 4 | Reserved | When read, the value after reset is read. |
| 3 | MBIST2ST | 0: MBIST2 passed successfully<br>1: MBIST2 has detected an error |
| 2 | MBIST1ST | 0: MBIST1 passed successfully<br>1: MBIST1 has detected an error |
| 1 | LBIST2ST | 0: LBIST2 passed successfully<br>1: LBIST2 has detected an error |
| 0 | LBIST1ST | 0: LBIST1 passed successfully<br>1: LBIST1 has detected an error |

Note 1. When the Field BIST is skipped, the value is shown as below.
[BSEQ0CTL.HWBISTEXE = 0 and System reset2]
[Debug mode(TRSTZ = 1)]
This register shows 0000 000F$_H$.
[Application reset1]
The previous value is kept.

### (16) BSEQ0CTL — Field BIST control register

This register is used to control the field BIST.

**Access:** This register can be Read/written in 32-bit units.

**Address:** FFF8 $0200_H$

**Value after reset:** 0000 $0001_H$ This register is reset by a Power On Reset or a System Reset1

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HWBISTEXE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 28.213 BSEQ0CTL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 0 | HWBISTEXE | The user can select by this bit if BIST shall be skip or not by next system reset2.<br>0: BIST is skipped by next system reset 2<br>1: BIST is executed |

## 28.6 ECM

The ECM monitors various failure detection states in the LSI chip, and defines the operation to be carried out upon failure detection. For the details of the ECM, see Section 29, Error Control Module (ECM).

## 28.7 CVM

The core voltage monitor (CVM) detects over and under voltage of the core voltage. For the details of the CVM, see **Section 10, Core Voltage Monitor (CVM)**.

## 28.8 CLMA

The clock monitors detect abnormal frequency of internal clocks related to the safety of the device. For the details of the CLMA, see **Section 13, Clock Monitor (CLMA)**.

## 28.9 WDTA

The window watchdog timer (WDTA) detects deadlock of CPU operation. For the details of the WDTA, see **Section 23, Window Watchdog Timer A (WDTA)**.

## 28.10 DCRB

The Data CRC Function B can be used to verify or generate CRC protected data streams of arbitrary length and different bit widths. For the details of the DCRB, see Section 30, Data CRC Function B (DCRB).

## 28.11 Difference among P1M-C, P1H-C and P1H-CE

ISO26262 is out of scope about P1H-CE.

**Table 28.214 Required ISO26262 ASIL on each device.**

| P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|
| ASIL-D Capable | ASIL-D Capable | ASIL-D Capable | N/A |

Difference among P1M-C and P1H-C registers can be referred in note of each table of list of registers.

# Section 29 Error Control Module (ECM)

## 29.1 Features of RH850/P1x-C Group ECM

### 29.1.1 Units

This microcontroller has the following number of ECM units.

Each ECM unit has one channel interface.

**Table 29.1 Unit Configurations and Channels**

| Unit Name (Channel Name) ECMn | Channels per Unit | Product | |
|---|---|---|---|
| | | P1M-C | P1H-C, P1H-CE |
| ECM0 | 1 | √ | √ |
| ECM1 | 1 | — | √ |

**Table 29.2 Index**

| Index | Meaning |
|---|---|
| n | Throughout this section, the individual ECM units are identified by the index “n” (n = 0, 1): for example, ECMnEPCFG is the ECM n error pulse configuration register. |
| m | Throughout this section, the individual ECM Master and ECM Checker are identified by the index “m” (m = M, C): for example, ECMmEST bit is the ECM m Error set trigger bit. |

### 29.1.2 Register Base Address

ECM base addresses are listed in the following table.

ECM register addresses are given as offsets from the base addresses in general.

**Table 29.3 Register Base Address**

| Base Address Name | Base Address | Supporting Device | |
|---|---|---|---|
| | | P1M-C | P1H-C, P1H-CE |
| <ECMM0_base> | FFD6 $0000_H$ | √ | √ |
| <ECMC0_base> | FFD6 $1000_H$ | √ | √ |
| <ECM0_base> | FFD6 $2000_H$ | √ | √ |
| <ECMM1_base> | FFCB $0000_H$ | — | √ |
| <ECMC1_base> | FFCB $1000_H$ | — | √ |
| <ECM1_base> | FFCB $2000_H$ | — | √ |

### 29.1.3 Clock Supply

Clock supply by and to ECM is listed in the following table.

**Table 29.4 Clock Supply**

| Unit Name | Clock for the Unit | Supply Clock Name |
|---|---|---|
| ECMn | P-Bus interface clock (PCLK) | CLK_HSB |
| | Delay timer clock (cntclk) | CLK_IOSC/2 |

### 29.1.4 Interrupt Requests

ECM interrupt requests are listed in the following table. The interrupt request signal is driven to the high level with a pulse width of one cycle of PCLK when error source status that interrupt generation is enabled is set.

**Table 29.5 Interrupt Requests**

| Unit Interrupt Name | Outline | Interrupt Number |
|---|---|---|
| INTECM0MI | ECM0 mask-able interrupt (EI level) | 0 |
| INTECMT0NMI | ECM0 non mask-able interrupt (FE level) | FEINT |
| INTECM1MI | ECM1 mask-able interrupt (EI level) | 0 |
| INTECMT1NMI | ECM1 non mask-able interrupt (FE level) | FEINT |

### 29.1.5 External Output Signals

External output signals of ECM are listed below.

**Table 29.6 External Output Signals**

| Unit Signal Name | Outline | Alternative port pin signal |
|---|---|---|
| ERROROUTZ | Error output signal | ERROROUTZ |

# 29.2 Overview

## 29.2.1 Specification Overview

ECM (Error Control Module) collects error signals coming from different error sources and monitoring circuits. It also outputs error signals from the error pins (ERROROUTZ) and generates interrupts and Error Control Module Reset signals. **Table 29.7** shows the specification overview of ECM.

**Table 29.7 Specification Overview**

| Item | Description |
|---|---|
| Safety processing | ECM can handle the following processing in response to error signal inputs from individual modules.<br>• Error flag set<br>• EI level interrupt generation<br>EI level interrupt generation can be controlled (enabled/disabled) for individual errors.<br>• FE level interrupt generation<br>FE level interrupt generation can be controlled (enabled/disabled) for individual errors.<br>• Internal reset generation<br>System reset 2 generation can be controlled (enabled/disabled) for individual errors.<br>• Error pin output<br>Pin output mask can be controlled (enabled/disabled) for individual errors.<br>Output can be toggled in response to a timer input or made at a fixed level. |
| Error status | ECM incorporates the error status register, which can be used to confirm the error status from the error flag.<br>The error flags are only cleared by software or an power on reset. In case of reset except for power on reset, the error flags are kept and the reset generation source can be confirmed by reading the status register after reset. |
| Debug, self-diagnosis | • Pseudo errors can be generated for debug and self-diagnosis.<br>The operation during injection of pseudo errors is identical to that for the occurrence of real errors. All configurations for the mask to the error pin output, interrupt, or Error Control Module Reset apply in the same way.<br>• ECM incorporates a loop-back function of the error pin output that is used to diagnose the path to the error output pin.<br>The status of the error output pin is reflected to an internal register and can be confirmed by reading the register. |
| Timeout function | ECM incorporates a function that generates an error signal output or Error Control Module Reset when the count value of the delay timer matches with the delay timer compare register because the delay timer was not stopped during the interrupt processing after being started simultaneously with the occurrence of an interrupt request. |
| Register protection | A write-protection with a special sequence is incorporated to protect registers from inadvertent write access. |
| ERROROUTZ clear masking | ECM incorporates a function that can mask software clearance for ERROROUTZ until the time which is counted from error occurrence reaches with the Error Output Clear Invalidation Configuration register. If another error occurs during time counting, then the time count is reset and restarted from the beginning. |
| Hi-Z control signal trigger | ECM triggers Hi-Z control signal into PIC.<br>Hi-Z control signal is triggered at the same condition when ECM activates ERROROUTZ pin.<br>Regarding of enabling of Hi-Z control, see **Section 26.4.4, Hi-Z control function over external pin for GTM output**. |
| Others | ECM is duplexed. ECM incorporates the error output pin. |

## 29.3 Block diagram

ECM0 and ECM1 are implemented redundant from ECM Master and ECM Checker. See **Figure 29.1** Connection of two ECM and **Figure 29.2** Connection among ECMn Master, ECMn Checker and peripherals.

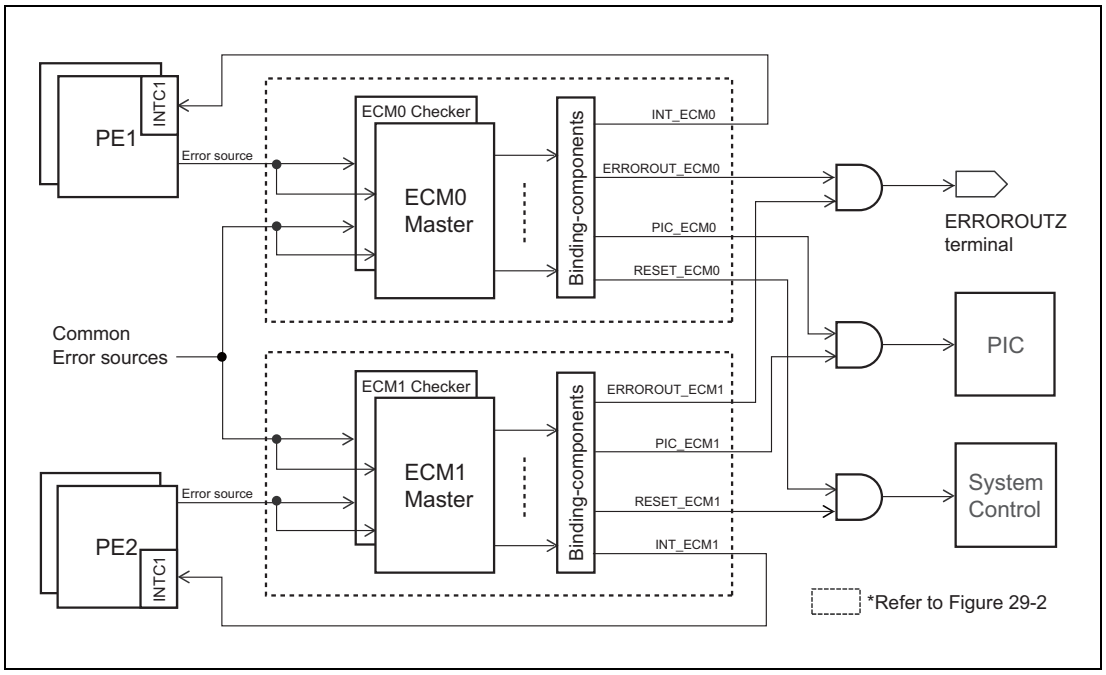



**Figure 29.1 Connection of two ECM**

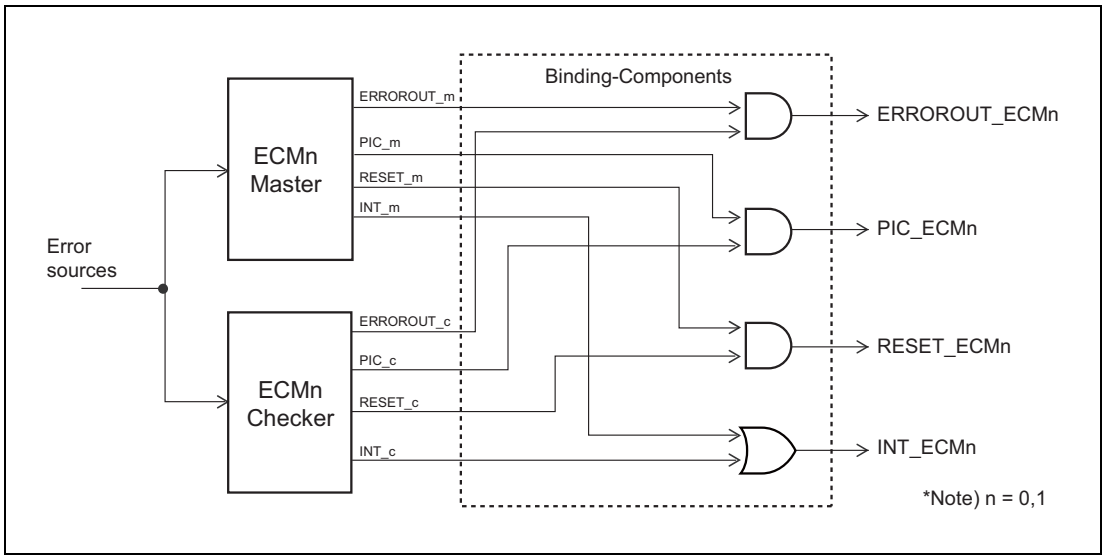



**Figure 29.2 Connection among ECMn Master, ECMn Checker and peripherals**

### 29.3.1 Error Input

**Table 29.8** shows the error inputs to ECM of RH850/P1x-C.

**Table 29.8 List of Error Inputs (1/10)**

| No. | Error sources | Error source explanation | Error type | | Product | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | PE related | Common | P1M-C | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
| 0 | Window watchdog timer for Main core and Sub core | WDTA0 error | √ | | √ | √ | √ | √ | √ |
| | | WDTA1 error | | | | | | | |
| 1 | LSDC Compare error | PE1 Redundant Lock-Step Compare Error | √ | | √ | √ | √ | √ | √ |
| | | PE2 Redundant Lock-Step Compare Error | | | | | | | |
| 2 | PFSS compare error - DMA, MECONT, IPIR | MECNT/IPIR Redundant Lock Step Compare Error | | √ | √ | √ | √ | √ | √ |
| | | DMA Redundant Lock Step Compare Error | | | | | | | |
| | | GRAM Write through buffer comparator Error | | | | | | | |
| 3 | Bus bridge error | FABT Arbitor Check Error | | √ | √ | √ | √ | √ | √ |
| | | GRAMC Arbitor Check Error Bank0 | | | | | | | |
| | | GRAMC Arbitor Check Error Bank1 | | | | | | | |
| | | GVCI Arbitor Check Error | | | | | | | |
| | | APB Arbitration Check Error | | | | | | | |
| | | APB Arbiter Compare Error | | | | | | | |

Table 29.8 List of Error Inputs (2/10)

| No. | Error sources | Error source explanation | Error type | | Product | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | PE related | Common | P1M-C | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
| 4 | Compare error of Safety Mechanism with redundancy | Compare error of FLI Address Parity BankA | | √ | √ | √ | √ | √ | √ |
| | | Compare error of FLI Address Parity BankB | | | | | | | |
| | | Compare error of VCI2CFB Data ECC | | | | | | | |
| | | Compare error of APB Register Address and Write Data ECC | | | | | | | |
| | | Compare error of AXI2GRAM GRAM Data ECC | | | | | | | |
| | | Compare error of GRAM Read Data ECC Data Path Bank0 | | | | | | | |
| | | Compare error of GRAM Read Data ECC Data Path Bank1 | | | | | | | |
| | | Compare error of VCI2GRAM GRAM Data ECC | | | | | | | |
| | | Compare error of GRAM Address ECC Bank0 Lower | | | | | | | |
| | | Compare error of GRAM Address ECC Bank0 Upper | | | | | | | |
| | | Compare error of GRAM Address ECC Bank1 Lower | | | | | | | |
| | | Compare error of GRAM Address ECC Bank1 Upper | | | | | | | |
| | | Compare error of VCI2CFB VCI Address ECC | | | | | | | |
| | | Compare error of VCI2API VCI Address ECC | | | | | | | |
| | | Compare error of MECNT/IPIR Redundant Lock Step Comparator | | | | | | | |
| | | Compare error of PE1 Redundant Lock Step Comparator | | | | | | | |
| | | Compare error of PE2 Redundant Lock Step Comparator | | | | | | | |
| | | Compare error of APB RmW Data ECC | | | | | | | |
| | | Compare error of APB ECC | | | | | | | |
| 5 | reserve | | | | | | | | |
| 6 | Temperature sensor error (over/under temp. detection) | Temperature abnormality error | | √ | √ | √ | √ | √ | √ |
| 7 | reserve | | | | | | | | |

Table 29.8 List of Error Inputs (3/10)

| No. | Error sources | Error source explanation | Error type | | Product | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | PE related | Common | P1M-C | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
| 8 | Clock monitor for Main OSC (CLMA0) | CLMA0 over frequency error | | √ | √ | √ | √ | √ | √ |
| | | CLMA0 under frequency error | | | | | | | |
| 9 | Clock monitor for WDT count clock (CLMA2) | CLMA2 over frequency error | | √ | √ | √ | √ | √ | √ |
| | | CLMA2 under frequency error | | | | | | | |
| 10 | Clock monitor for CPU clock (CLMA3/CLMA4)*2 | CLMA3 over frequency error | √ | | √ | √ | √ | √ | √ |
| | | CLMA3 under frequency error | | | | | | | |
| | | CLMA4 over frequency error | | | | | | | |
| | | CLMA4 under frequency error | | | | | | | |
| 11 | Clock monitor for ICUMC clock (CLMA5) | CLMA5 over frequency error | | √ | √ | √ | √ | √ | √ |
| | | CLMA5 under frequency error | | | | | | | |
| 12 | Clock monitor for Peripheral clock (CLMA1) | CLMA1 over frequency error | | √ | √ | √ | √ | √ | √ |
| | | CLMA1 under frequency error | | | | | | | |
| 13 | reserve | | | | | | | | |
| 14 | reserve | | | | | | | | |
| 15 | reserve | | | | | | | | |
| 16 | Local RAM ECC - uncorrectable error | PE1 Local RAM Data ECC DED | √ | | √ | √ | √ | √ | √ |
| | | PE2 Local RAM Data ECC DED | | | | | | | |
| 17 | Global RAM ECC - uncorrectable error | GRAM Address ECC DED | | √ | √ | √ | √ | √ | √ |
| | | GRAM Data ECC DED | | | | | | | |
| 18 | Instruction Cache RAM EDC - uncorrectable error | PE1 F-Cache Data RAM Data ECC DED | √ | | √ | √ | √ | √ | √ |
| | | PE1 F-Cache Data RAM Data ECC SED | | | | | | | |
| | | PE1 F-Cache Tag RAM Data ECC DED | | | | | | | |
| | | PE1 F-Cache Tag RAM Data ECC SED | | | | | | | |
| | | PE2 F-Cache Data RAM Data ECC DED | | | | | | | |
| | | PE2 F-Cache Data RAM Data ECC SED | | | | | | | |
| | | PE2 F-Cache Tag RAM Data ECC DED | | | | | | | |
| | | PE2 F-Cache Tag RAM Data ECC SED | | | | | | | |
| 19 | Code Flash ECC - uncorrectable error - Code flash address parity error | FLI Data ECC DED | | √ | √ | √ | √ | √ | √ |
| | | FLI Address Parity Error | | | | | | | |
| 20 | Data Flash ECC - uncorrectable error | Data Flash0 ECC DED | | √ | √ | √ | √ | √ | √ |
| | | Data Flash1 ECC DED | | | | | | | |

Table 29.8 List of Error Inputs (4/10)

| No. | Error sources | Error source explanation | Error type | | Product | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | PE related | Common | P1M-C | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
| 21 | Peripheral (CSIH: SPI) RAM ECC - uncorrectable error | CSIH0 RAM ECC 2bit error | | √ | √ | √ | √ | √ | √ |
| | | CSIH1 RAM ECC 2bit error | | | | | | | |
| | | CSIH2 RAM ECC 2bit error | | | | | | | |
| | | CSIH3 RAM ECC 2bit error | | | | | | | |
| 22 | Peripheral (MCAN) RAM ECC - uncorrectable error | MTTCAN0 RAM ECC 2bit error | | √ | √ | √ | √ | √ | √ |
| | | MCAN0 RAM ECC 2bit error | | | | | | | |
| | | MCAN1 RAM ECC 2bit error | | | | | | | |
| | | MCAN2 RAM ECC 2bit error | | | | | | | |
| 23 | Peripheral (Ethernet) RAM ECC - uncorrectable error | ETC0 TNRAM ECC 2bit error | | √ | √ | √ | √ | √ | √ |
| | | ETC0 RNRAM ECC 2bit error | | | | | | | |
| | | ETC0 RETRYRAM ECC 2bit error | | | | | | | |
| | | ETC1 TNRAM ECC 2bit error | | | | | | | |
| | | ETC1 RNRAM ECC 2bit error | | | | | | | |
| | | ETC1 RETRYRAM ECC 2bit error | | | | | | | |
| 24 | Peripheral (FlexRay) RAM ECC - uncorrectable error | FRAY0 RAM0 ECC 2bit error | | √ | √ | √ | √ | √ | √ |
| | | FRAY0 RAM1 ECC 2bit error | | | | | | | |
| | | FRAY0 RAM2 ECC 2bit error | | | | | | | |
| | | FRAY1 RAM0 ECC 2bit error | | | | | | | |
| | | FRAY1 RAM1 ECC 2bit error | | | | | | | |
| | | FRAY1 RAM2 ECC 2bit error | | | | | | | |
| 25 | Peripheral (GTM) RAM ECC - uncorrectable error | GTM RAM0 ECC 2bit error | | √ | √ | √ | √ | √ | √ |
| | | GTM RAM1 ECC 2bit error | | | | | | | |
| | | GTM RAM2 ECC 2bit error | | | | | | | |
| | | GTM RAM3 ECC 2bit error | | | | | | | |
| 26 | reserve | | | | | | | | |
| 27 | reserve | | | | | | | | |

Table 29.8 List of Error Inputs (5/10)

| No. | Error sources | Error source explanation | Error type | | Product | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | PE related | Common | P1M-C | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
| 28 | Bus ECC error: DED | APB Register Address ECC DED | | √ | √ | √ | √ | √ | √ |
| | | APB Register Write Data ECC DED | | | | | | | |
| | | VCI2CFB VCI Address ECC DED | | | | | | | |
| | | VCI2API VCI Address ECC DED | | | | | | | |
| | | PE1 IFU Data ECC DED | | | | | | | |
| | | PE2 IFU Data ECC DED | | | | | | | |
| | | PE1 LVCIM Data ECC DED | | | | | | | |
| | | PE2 LVCIM Data ECC DED | | | | | | | |
| | | PE1 LVCIS Address ECC DED | | | | | | | |
| | | PE1 LVCIS Data ECC DED | | | | | | | |
| | | PE2 LVCIS Address ECC DED | | | | | | | |
| | | PE2 LVCIS Data ECC DED | | | | | | | |
| | | DMA Transfer Data ECC DED | | | | | | | |
| | | APB RmW Data ECC DED | | | | | | | |
| | | APB Address ECC DED | | | | | | | |
| | | APB Data ECC DED | | | | | | | |

Table 29.8 List of Error Inputs (6/10)

| No. | Error sources | Error source explanation | Error type | | Product | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | PE related | Common | P1M-C | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
| 29 | Bus ECC error: SED | APB Register Address ECC SED | | √ | √ | √ | √ | √ | √ |
| | | APB Register Write Data ECC SED | | | | | | | |
| | | VCI2CFB VCI Address ECC SED | | | | | | | |
| | | VCI2API VCI Address ECC SED | | | | | | | |
| | | PE1 IFU Data ECC SED | | | | | | | |
| | | PE2 IFU Data ECC SED | | | | | | | |
| | | PE1 LVCIM Data ECC SED | | | | | | | |
| | | PE2 LVCIM Data ECC SED | | | | | | | |
| | | PE1 LVCIS Address ECC SED | | | | | | | |
| | | PE1 LVCIS Data ECC SED | | | | | | | |
| | | PE2 LVCIS Address ECC SED | | | | | | | |
| | | PE2 LVCIS Data ECC SED | | | | | | | |
| | | DMA Transfer Data ECC SED | | | | | | | |
| | | APB RmW Data ECC SED | | | | | | | |
| | | APB Address ECC SED | | | | | | | |
| | | APB Data ECC SED | | | | | | | |
| 30 | reserve | | | | | | | | |
| 31 | reserve | | | | | | | | |
| 32 | Local RAM error address overflow | PE1 Local RAM Data ECC Error Count Overflow | √ | | √ | √ | √ | √ | √ |
| | | PE2 Local RAM Data ECC Error Count Overflow | | | | | | | |
| 33 | Global RAM error address overflow | GRAM Data ECC Error Count Overflow | | √ | √ | √ | √ | √ | √ |
| 34 | reserve | | | | | | | | |
| 35 | Code Flash ECC error address overflow | FLI Data ECC Error Overflow | | √ | √ | √ | √ | √ | √ |
| 36 | Data Flash error ECC error address overflow | Data Flash0 ECC Error Overflow | | √ | √ | √ | √ | √ | √ |
| | | Data Flash1 ECC Error Overflow | | | | | | | |

Table 29.8 List of Error Inputs (7/10)

| No. | Error sources | Error source explanation | Error type | | Product | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | PE related | Common | P1M-C | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
| 37 | Peripheral RAM ECC error address overflow | CSIH0 RAM ECC error address overflow | | √ | √ | √ | √ | √ | √ |
| | | CSIH1 RAM ECC error address overflow | | | | | | | |
| | | CSIH2 RAM ECC error address overflow | | | | | | | |
| | | CSIH3 RAM ECC error address overflow | | | | | | | |
| | | MTTCAN0 RAM ECC error address overflow | | | | | | | |
| | | MCAN0 RAM ECC error address overflow | | | | | | | |
| | | MCAN1 RAM ECC error address overflow | | | | | | | |
| | | MCAN2 RAM ECC error address overflow | | | | | | | |
| | | FRAY0 RAM0 ECC error address overflow | | | | | | | |
| | | FRAY0 RAM1 ECC error address overflow | | | | | | | |
| | | FRAY0 RAM2 ECC error address overflow | | | | | | | |
| | | FRAY1 RAM0 ECC error address overflow | | | | | | | |
| | | FRAY1 RAM1 ECC error address overflow | | | | | | | |
| | | FRAY1 RAM2 ECC error address overflow | | | | | | | |
| | | GTM RAM0 ECC error address overflow | | | | | | | |
| | | GTM RAM1 ECC error address overflow | | | | | | | |
| | | GTM RAM2 ECC error address overflow | | | | | | | |
| | | GTM RAM3 ECC error address overflow | | | | | | | |
| 38 | reserve | | | | | | | | |
| 39 | reserve | | | | | | | | |
| 40 | DTSRAM ECC - uncorrectable error | DTS RAM Data ECC DED | | √ | √ | √ | √ | √ | √ |
| 41 | DTSRAM ECC - correctable error*1 | DTS RAM Data ECC SED | | √ | √ | √ | √ | √ | √ |
| 42 | reserve | | | | | | | | |
| 43 | reserve | | | | | | | | |
| 44 | reserve | | | | | | | | |
| 45 | reserve | | | | | | | | |
| 46 | reserve | | | | | | | | |
| 47 | reserve | | | | | | | | |
| 48 | Local RAM ECC - correctable error*1 | PE1 Local RAM Data ECC SED | √ | | √ | √ | √ | √ | √ |
| | | PE2 Local RAM Data ECC SED | | | | | | | |
| 49 | Global RAM ECC - correctable error*1 | GRAM Address ECC SED | | √ | √ | √ | √ | √ | √ |
| | | GRAM Data ECC SED | | | | | | | |
| 50 | reserve | | | | | | | | |

Table 29.8 List of Error Inputs (8/10)

| No. | Error sources | Error source explanation | Error type | | Product | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | PE related | Common | P1M-C | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
| 51 | Code Flash ECC - correctable error*1 | FLI Data ECC SED | | √ | √ | √ | √ | √ | √ |
| 52 | Data Flash ECC - correctable error*1 | Data Flash0 ECC SED | | √ | √ | √ | √ | √ | √ |
| | | Data Flash1 ECC SED | | | | | | | |
| 53 | Peripheral (CSIH: SPI) RAM ECC - correctable error*1 | CSIH0 RAM ECC 1bit error | | √ | √ | √ | √ | √ | √ |
| | | CSIH1 RAM ECC 1bit error | | | | | | | |
| | | CSIH2 RAM ECC 1bit error | | | | | | | |
| | | CSIH3 RAM ECC 1bit error | | | | | | | |
| 54 | Peripheral (MCAN) RAM ECC - correctable error*1 | MTTCAN0 RAM ECC 1bit error | | √ | √ | √ | √ | √ | √ |
| | | MCAN0 RAM ECC 1bit error | | | | | | | |
| | | MCAN1 RAM ECC 1bit error | | | | | | | |
| | | MCAN2 RAM ECC 1bit error | | | | | | | |
| 55 | Peripheral (Ethernet) RAM ECC - correctable error*1 | ETC0 TNRAM ECC 1bit error | | √ | √ | √ | √ | √ | √ |
| | | ETC0 RNRAM ECC 1bit error | | | | | | | |
| | | ETC0 RETRYRAM ECC 1bit error | | | | | | | |
| | | ETC1 TNRAM ECC 1bit error | | | | | | | |
| | | ETC1 RNRAM ECC 1bit error | | | | | | | |
| | | ETC1 RETRYRAM ECC 1bit error | | | | | | | |
| 56 | Peripheral (FlexRay) RAM ECC - correctable error*1 | FRAY0 RAM0 ECC 1bit error | | √ | √ | √ | √ | √ | √ |
| | | FRAY0 RAM1 ECC 1bit error | | | | | | | |
| | | FRAY0 RAM2 ECC 1bit error | | | | | | | |
| | | FRAY1 RAM0 ECC 1bit error | | | | | | | |
| | | FRAY1 RAM1 ECC 1bit error | | | | | | | |
| | | FRAY1 RAM2 ECC 1bit error | | | | | | | |
| 57 | Peripheral (GTM) RAM ECC - correctable error*1 | GTM RAM0 ECC 1bit error | | √ | √ | √ | √ | √ | √ |
| | | GTM RAM1 ECC 1bit error | | | | | | | |
| | | GTM RAM2 ECC 1bit error | | | | | | | |
| | | GTM RAM3 ECC 1bit error | | | | | | | |
| 58 | reserve | | | | | | | | |
| 59 | reserve | | | | | | | | |
| 60 | reserve | | | | | | | | |
| 61 | reserve | | | | | | | | |
| 62 | reserve | | | | | | | | |

Table 29.8 List of Error Inputs (9/10)

| No. | Error sources | Error source explanation | Error type | | Product | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | PE related | Common | P1M-C | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
| 63 | reserve | | | | | | | | |
| 64 | PE guard error | PE1 PEG Error | √ | | √ | √ | √ | √ | √ |
| | | PE2 PEG Error | | | | | | | |
| 65 | Global RAM guard error | AXI2GRAM GRAM Protection Error | | √ | √ | √ | √ | √ | √ |
| | | PE1 GRAM Protection Error | | | | | | | |
| | | PE2 GRAM Protection Error | | | | | | | |
| | | VCI2GRAM GRAM Protection Error | | | | | | | |
| 66 | MEMC guard error | memc guard error | | √ | N/A | √ | N/A | √ | √ |
| 67 | slave guard error | GUARD APB Register Guard Error | | √ | √ | √ | √ | √ | √ |
| | | hbss slave guard error | | | | | | | |
| | | APB slave guard error | | | | | | | |
| 68 | reserved | | | | | | | | |
| 69 | reserved | | | | | | | | |
| 70 | reserved | | | | | | | | |
| 71 | reserved | | | | | | | | |
| 72 | Illegal address error - Code Flash illegal access by PE1/PE2 | PE1 FLI Illegal Area Access Error | √ | | √ | √ | √ | √ | √ |
| | | PE2 FLI Illegal Area Access Error | | | | | | | |
| 73 | Illegal address error - Global RAM illegal access by PE1/PE2 | PE1 GRAM Illegal Area Access Error | √ | | √ | √ | √ | √ | √ |
| | | PE2 GRAM Illegal Area Access Error | | | | | | | |
| 74 | Illegal address error - LPB illegal access by PE1/PE2 | PE1 LAPB Illegal Area Access Error | √ | | √ | √ | √ | √ | √ |
| | | PE2 LAPB Illegal Area Access Error | | | | | | | |
| 75 | Illegal address error - PBUS illegal access | APB Illegal Area Access Error | | √ | √ | √ | √ | √ | √ |
| 76 | Illegal address error - HBUS illegal access | HBSS illegal access by PE1,2 | | √ | √ | √ | √ | √ | √ |
| 77 | Illegal address error - Code Flash illegal access via GVCI | VCI2CFB FLI Illegal Area Access Error | | √ | √ | √ | √ | √ | √ |
| 78 | Illegal address error - Global RAM illegal access via GVCI/Hbus | AXI2GRAM GRAM Illegal Area Access Error | | √ | √ | √ | √ | √ | √ |
| | | VCI2GRAM GRAM Illegal Area Access Error | | | | | | | |
| 79 | Illegal address error - Reserved Area illegal access from Hbus | HBSS illegal access by DMA | | √ | √ | √ | √ | √ | √ |
| 80 | DMA transfer error | DMA Transfer Error | | √ | √ | √ | √ | √ | √ |
| 81 | DMA illegal access error | DMA Register Access Violation | | √ | √ | √ | √ | √ | √ |
| 82 | Flash - Flash sequencer error | Flash0 sequencer error | | √ | √ | √ | √ | √ | √ |
| | | Flash1 sequencer error | | | | | | | |

Table 29.8 List of Error Inputs (10/10)

| No. | Error sources | Error source explanation | Error type | | Product | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | PE related | Common | P1M-C | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | P1H-CE |
| 83 | Flash<br>- ECC correctable/ uncorrectable errors during FACI reset/refresh transfer | FACI Reset/Refresh transfer error | | √ | √ | √ | √ | √ | √ |
| 84 | A/D Converter parity error | ADCE0 Parity Error | | √ | √ | √ | √ | √ | √ |
| | | ADCE1 Parity Error | | | | | | | |
| 85 | reserved | | | | | | | | |
| 86 | reserved | | | | | | | | |
| 87 | Unintended DISABLE/ ENABLE of PE2 | This error is issued when control signal of “PE2 disable” causes parity error. | | √ | | √ | √ | √ | √ |
| 88 | Mode error<br>- Unintended deactivation of user mode | This error is issued when operating mode of the device becomes other than user mode regardless of mode pin setting is user mode.*3 | | √ | √ | √ | √ | √ | √ |
| 89 | Mode error<br>- Unintended activation of Code Flash Programming mode | This error is issued when operating mode of the device becomes serial programming mode regardless of mode pin setting is normal operating mode.*3 | | √ | √ | √ | √ | √ | √ |
| 90 | Mode error<br>- Unintended Debug Enable detection | This error is issued when CPU operating mode (PE1) transits to debug mode without authentication from debugger.*4 | √ | | √ | √ | √ | √ | √ |
| | | This error is issued when CPU operating mode (PE2) transits to debug mode without authentication from debugger.*4 | | | | | | | |
| 91 | Mode error<br>- Unintended activation of Test Mode | This error is issued when operating mode of the device becomes test mode regardless of mode pin setting is normal operating mode.*3 | | √ | √ | √ | √ | √ | √ |
| 92 | ECM compare error | ECM0 compare error | √ | | √ | √ | √ | √ | √ |
| | | ECM1 compare error | | | | | | | |

Note 1. If correction is disabled, single bit errors are still reported as correctable errors to the ECM

Note 2. CLMA3 supervises PE1 clock. CLMA4 supervises PE2 clock. P1M-C device doesn’t have CLMA4.

Note 3. For details of regarding each mode, see the **Section 5, Operating Modes**.

Note 4. For details of regarding debug mode, see the **Section 31, On-Chip Debugging Unit (OCD)**.

### 29.3.2 Operations for Error Output

After reset release, the ERROROUTZ pin outputs the low (error) level. Follow the procedure described in **Section 29.4.3, ECMmnECLR — ECM Master/Checker n Error Clear Trigger Register (m = M/C)**, to clear the error before using ECM.

The error output can be configured for two different modes of operation, non-dynamic or dynamic.

| Error Status ECMmSSE031 to ECMmSSE000 ECMmSSE131 to ECMmSSE100 ECMmSSE230 to ECMmSSE200 | Operating Mode ECMSL0 Bit | Error Output Operating Mode | Error Output Level | Error Status |
|---|---|---|---|---|
| 0 | 0 | Non-dynamic | H | No error |
| | 1 | Dynamic | Toggles (according to timer input) | No error |
| 1 | 0 | Non-dynamic | L | Error |
| | 1 | Dynamic | L | Error |

### 29.3.3 ERROROUTZ behavior at reset

Below table explains the behavior of the error output logic and the ERROROUTZ pin at reset. Also the level of the ERROROUTZsignal during and after reset is explained.

**Table 29.9 ERROROUTZ behavior at reset**

| | Reset signal for initialize | | | |
|---|---|---|---|---|
| Category | Power on reset | System reset 1 | System reset 2 | Application reset 1 |
| Error Pin Logic | √ | √ | √ | — |
| Error Pin Buffer | √ | √ | √ | — |
| ERROROUT pin level during reset | Hi-Z | Hi-Z | Low level | — |
| ERROROUT pin level after reset | Low level | Low level | Low level | Level according to error status before reset |

#### 29.3.3.1 Dynamic Mode Enable

1. Initialize the related timer GTMAT0O5.
2. Set the error output to high level by setting the ECMmECT (m = M/C) bit in the ECM master/checker error clear trigger register to 1.
3. Set the ECMnEPCFG.ECMSL0 bit to 1 for dynamic mode.
4. Start the timer GTMAT0O5.

#### 29.3.3.2 Dynamic Mode Disable

1. Set the error output to low level by setting the ECMmEST bit (m = M/C) in the ECM master/checker error set trigger register to 1.
2. Stop the timer GTMAT0O5.
3. Clear the ECMSL0 bit in the ECM error pulse configuration register to 0 to specify non-dynamic mode.

### 29.3.4 Error Status

The error status is indicated by ECM master/checker error source status register 0 and ECM master/checker error source status register 1 and ECM master/checker error source status register 2. The error status is only cleared by software or a power on reset. In case of reset except for power on reset, the error status is kept and the error of the reset source can be confirmed by reading the ECM master/checker error source status register 0 and ECM master/checker error source status register 1 and ECM master/checker error source status register 2 after reset release.

### 29.3.5 Writing to Protected Registers

Write protected registers are protected from inadvertent write access due to erroneous program execution, etc. An overview of the related registers is shown in **Section 29.4, Register Specification**, Overview of ECM Registers. In this context all ECM master registers belong to the protection command register ECMMnPCMD0, while the ECM checker registers belong to ECMCnPCMD0. The ECM common registers are protected by ECMnPCMD1.

#### 29.3.5.1 Protection Unlock Sequence

Write access to a write protected register is only possible within a special protection unlock sequence.

1. Write the fixed value 0000 $00A5_H$ to the protection command registers ECMnPCMD1 or ECMmnPCMD0.
2. Write the desired value to the registers protected by ECMnPCMD1 and ECMmnPCMD0.
3. Write the bitwise inversion of desired value to the registers protected by ECMnPCMD1 or ECMmnPCMD0.
4. Write the desired value to the registers protected by ECMnPCMD1 or ECMmnPCMD0.
5. Check successful write of the desired value to the protected register by checking that ECMnPS.ECMPRERR = 0.

In case of any access to another register between step 1 to step 4 of the above sequence, the protection mechanism behaves as follows:

- If the second register belongs to the ECM, the write to the protected register fails (indicated by ECMnPS.ECMPRERR = 1). The entire sequence has to be restarted at step 1.
- If the second register does not belong to the ECM, the protection unlock sequence is not disrupted and the write to the first register can be completed successfully.

**NOTE**

Note: For sequences of ECMnPCMD1 and ECMmnPCMD0 registers, the status of the protection unlock sequence is commonly indicated by the ECMnPS.ECMPRERR flag. Therefore, it is recommended not to intermix protection unlock sequences of ECMnPCMD1 and ECMmnPCMD0.

In case the protection unlock sequence is interrupted, the protection mechanism behaves as follows:

- Interrupts during protection unlock sequence
  If an interrupt is acknowledged within the above protection unlock sequence and the interrupt service routine does not access any register of the ECM, the protection unlock sequence is not

disrupted and the write to the protected register can be successfully completed after returning from the interrupt service routine.

**CAUTION**

**All protected registers as well as the ECM protection command registers ECMnPCMD1 and ECMmnPCMD0 must be accessed in 32-bit units.**

### 29.3.6 Timeout Function for Interrupt Processing

The delay timer incorporated to ECM can be started simultaneously with the occurrence of an interrupt request. ECM incorporates a function that generates an error signal output or Error Control Module Reset when the count value of the delay timer matches with the value of the delay timer compare register because the delay timer was not stopped during the interrupt processing. The timer counting is not stopped when a break occurs.

The counting of the delay timer always starts from 0. Configure the duration until a Error Control Module Reset or error output is generated with the settings of the delay timer compare register.

### 29.3.7 Configuration lock

The following configuration registers in ECM are protected by a special sequence and slave guard.

- Error Set Trigger Register (**Section 29.4.2**)
- Error Clear Trigger Register (**Section 29.4.3**)
- Error Pulse Configuration Register (**Section 29.4.8**)
- Interrupt configuration Register (**Section 29.4.9**, **Section 29.4.10**, **Section 29.4.11**, **Section 29.4.12**, **Section 29.4.13**, **Section 29.4.14**)
- Internal Reset configuration Register (**Section 29.4.15**, **Section 29.4.16**, **Section 29.4.17**)
- Error Mask Register (**Section 29.4.18**, **Section 29.4.19**, **Section 29.4.20**)
- Error Source Status Clear Trigger Register (**Section 29.4.21**, **Section 29.4.22**, **Section 29.4.23**)
- Pseudo Error Trigger Register (**Section 29.4.26**, **Section 29.4.27**, **Section 29.4.28**)
- Delay Timer Control Register (**Section 29.4.29**)
- Delay Timer Compare Register (**Section 29.4.31**)
- Delay Timer Configuration Register (**Section 29.4.32**, **Section 29.4.33**, **Section 29.4.34**, **Section 29.4.35**, **Section 29.4.36**, **Section 29.4.37**)
- Error Output Clear Invalidation Configuration Register (**Section 29.4.38**)

The detail of special protection sequence is described in **Section 29.3.5, Writing to Protected Registers.**

Slave guard is described in **Section 28, Functional Safety**.

### 29.3.8 Masking of “Error clear trigger register”

The active error output status must be cleared by software via the Error clear trigger register (ECMMECLR/ECMCECLR). A minimum activation time of the error output is achieved by the Error output clear invalidation counter. This counter is (re)started each time a new error event is triggered at the ECM. It counts up from $0000_H$ to $FFFF_H$. Error output clear by software is not possible unless this counter reaches the compare value configured in the ECMnEOCCFG register. If Error output clear invalidation counter is still running, Error output clear is masked and Error output clear request by software is not memorized.

# 29.4 Register Specification

## 29.4.1 List of Registers

ECM consists of three address areas: common part, ECM master, and ECM checker.

The following shows the register map of the ECM master and checker registers.

**Table 29.10 Address List of ECM Master and Checker Registers**

| Address | Register Symbol | Register Name | R/W | Access Width | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|---|
| | | | | | | PBG | Protection by Sequence |
| **ECM Master Registers** | | | | | | **<ECMM0_base: FFD6 0000$_H$>** **<ECMM1_base: FFCB 0000$_H$>** | |
| <ECMMn_base> | ECMMnESET | ECM master n error set trigger register | W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMMn_base> + 04$_H$ | ECMMnECLR | ECM master n error clear trigger register | W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMMn_base> + 08$_H$ | ECMMnESSTR0 | ECM master n error source status register 0 | R | 32 | 0000 0000$_H$ | *1 | Not protected |
| <ECMMn_base> + 0C$_H$ | ECMMnESSTR1 | ECM master n error source status register 1 | R | 32 | 0000 0000$_H$ | *1 | Not protected |
| <ECMMn_base> + 10$_H$ | ECMMnESSTR2 | ECM master n error source status register 2 | R | 32 | 0000 0000$_H$ | *1 | Not protected |
| <ECMMn_base> + 14$_H$ | ECMMnPCMD0 | ECM master n protection command register | W | 32 | Undefined | *1 | Not protected |
| **ECM Checker Registers** | | | | | | **<ECMC0_base: FFD6 1000$_H$>** **<ECMC1_base: FFCB 1000$_H$>** | |
| <ECMCn_base> | ECMCnESET | ECM checker n error set trigger register | W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMCn_base> + 04$_H$ | ECMCnECLR | ECM checker n error clear trigger register | W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMCn_base> + 08$_H$ | ECMCnESSTR0 | ECM checker n error source status register 0 | R | 32 | 0000 0000$_H$ | *1 | Not protected |
| <ECMCn_base> + 0C$_H$ | ECMCnESSTR1 | ECM checker n error source status register 1 | R | 32 | 0000 0000$_H$ | *1 | Not protected |
| <ECMCn_base> + 10$_H$ | ECMCnESSTR2 | ECM checker n error source status register 2 | R | 32 | 0000 0000$_H$ | *1 | Not protected |
| <ECMCn_base> + 14$_H$ | ECMCnPCMD0 | ECM checker n protection command register | W | 32 | Undefined | *1 | Not protected |

Note 1. ECMM/C0_base Registers:PBG3#0.PG3-ECM0
ECMM/C1_base Registers:PBG1#0.PG1-ECM1

The following shows the register map of the ECM common part.

Table 29.11 Address List of ECM common Registers (1/2)

| | | | | | <ECM0_base: FFD6 2000$_H$><br><ECM1_base: FFCB 2000$_H$> | | |
|---|---|---|---|---|---|---|---|
| | | | | | | Access Protection | |
| Address | Register Symbol | Register Name | R/W | Access Width | Value after reset | PBG | Protection by Sequence |
| <ECMn_base> | ECMnEPCFG | ECM n error pulse configuration register | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 04$_H$ | ECMnMICFG0 | ECM n maskable interrupt configuration register 0 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 08$_H$ | ECMnMICFG1 | ECM n maskable interrupt configuration register 1 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 0C$_H$ | ECMnMICFG2 | ECM n maskable interrupt configuration register 2 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 10$_H$ | ECMnNMICFG0 | ECM n non-maskable interrupt configuration register 0 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 14$_H$ | ECMnNMICFG1 | ECM n non-maskable interrupt configuration register 1 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 18$_H$ | ECMnNMICFG2 | ECM n non-maskable interrupt configuration register 2 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 1C$_H$ | ECMnIRCFG0 | ECM n internal reset configuration register 0 | R/W | 32 | 0000 0001$_H$ | *1 | Protected |
| <ECMn_base> + 20$_H$ | ECMnIRCFG1 | ECM n internal reset configuration register 1 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 24$_H$ | ECMnIRCFG2 | ECM n internal reset configuration register 2 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 28$_H$ | ECMnEMK0 | ECM n error mask register 0 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 2C$_H$ | ECMnEMK1 | ECM n error mask register 1 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 30$_H$ | ECMnEMK2 | ECM n error mask register 2 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 34$_H$ | ECMnESSTC0 | ECM n error source status clear register 0 | W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 38$_H$ | ECMnESSTC1 | ECM n error source status clear register 1 | W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 3C$_H$ | ECMnESSTC2 | ECM n error source status clear register 2 | W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 40$_H$ | ECMnPCMD1 | ECM n protection command register | W | 32 | Undefined | *1 | Not protected |
| <ECMn_base> + 44$_H$ | ECMnPS | ECM n protection status register | R | 8 | 00$_H$ | *1 | Not protected |
| <ECMn_base> + 48$_H$ | ECMnPE0 | ECM n pseudo error trigger register 0 | W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 4C$_H$ | ECMnPE1 | ECM n pseudo error trigger register 1 | W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 50$_H$ | ECMnPE2 | ECM n pseudo error trigger register 2 | W | 32 | 0000 0000$_H$ | *1 | Protected |

Table 29.11 Address List of ECM common Registers (2/2)

| | | | | | <ECM0_base: FFD6 2000$_H$><br><ECM1_base: FFCB 2000$_H$> | | |
|---|---|---|---|---|---|---|---|
| | | | | | | Access Protection | |
| Address | Register Symbol | Register Name | R/W | Access Width | Value after reset | PBG | Protection by Sequence |
| <ECMn_base> + 54$_H$ | ECMnDTMCTL | ECM n delay timer control register | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 58$_H$ | ECMnDTMR | ECM n delay timer register | R | 16 | 0000$_H$ | *1 | Not protected |
| <ECMn_base> + 5C$_H$ | ECMnDTMCMP | ECM n delay timer compare register | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 60$_H$ | ECMnDTMCFG0 | ECM n delay timer configuration register 0 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 64$_H$ | ECMnDTMCFG1 | ECM n delay timer configuration register 1 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 68$_H$ | ECMnDTMCFG2 | ECM n delay timer configuration register 2 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 6C$_H$ | ECMnDTMCFG3 | ECM n delay timer configuration register 3 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 70$_H$ | ECMnDTMCFG4 | ECM n delay timer configuration register 4 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 74$_H$ | ECMnDTMCFG5 | ECM n delay timer configuration register 5 | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 78$_H$ | ECMnEOCCFG | ECM n error output clear invalidation configuration register | R/W | 32 | 0000 0000$_H$ | *1 | Protected |
| <ECMn_base> + 7C$_H$ | ECMnPEM | ECM n pseudo error mask register | R/W | 32 | 0000 0000$_H$ | *1 | Not protected |

Note 1. ECM0_base Registers:PBG3#0.PG3-ECM0
ECM1_base Registers:PBG1#0.PG1-ECM1

The ECM registers are the register areas common to the redundancy area to be implemented. Writes to the common register areas are conducted simultaneously. The common area for ECM master is read by reading access to the common area. The ECM master register and the ECM checker register represent the address areas which can be written separately.

ECM Register reset condition is shown in **Table 29.12** and **Table 29.13**.

Table 29.12 Reset condition of ECM Master and Checker Registers (1/2)

| | | Reset condition | | | | |
|---|---|---|---|---|---|---|
| Register Symbol | Register Name | Power On Reset | System Reset1 | System Reset2 | Application Reset1 | Limited Reset |
| **ECM Master Registers** | | | | | | |
| ECMMnESET | ECM master n error set trigger register | √ | √ | √ | √ | — |
| ECMMnECLR | ECM master n error clear trigger register | √ | √ | √ | √ | — |
| ECMMnESSTR0 | ECM master n error source status register 0 | √ | — | — | — | — |

Table 29.12 Reset condition of ECM Master and Checker Registers (2/2)

| Register Symbol | Register Name | Reset condition | | | | |
|---|---|---|---|---|---|---|
| | | Power On Reset | System Reset1 | System Reset2 | Application Reset1 | Limited Reset |
| ECMMnESSTR1 | ECM master n error source status register 1 | √ | — | — | — | — |
| ECMMnESSTR2 | ECM master n error source status register 2 | √ | — | — | — | — |
| ECMMnPCMD0 | ECM master n protection command register | √ | √ | √ | √ | — |
| **ECM Checker Registers** | | | | | | |
| ECMCnESET | ECM checker n error set trigger register | √ | √ | √ | √ | — |
| ECMCnECLR | ECM checker n error clear trigger register | √ | √ | √ | √ | — |
| ECMCnESSTR0 | ECM checker n error source status register 0 | √ | — | — | — | — |
| ECMCnESSTR1 | ECM checker n error source status register 1 | √ | — | — | — | — |
| ECMCnESSTR2 | ECM checker n error source status register 2 | √ | — | — | — | — |
| ECMCnPCMD0 | ECM checker n protection command register | √ | √ | √ | √ | — |

Table 29.13 Reset condition of ECM Common Registers (1/2)

| Register Symbol | Register Name | Reset condition | | | | |
|---|---|---|---|---|---|---|
| | | Power On Reset | System Reset1 | System Reset2 | Application Reset1 | Limited Reset |
| ECMnEPCFG | ECM n error pulse configuration register | √ | √ | √ | √ | — |
| ECMnMICFG0 | ECM n maskable interrupt configuration register 0 | √ | √ | √ | √ | — |
| ECMnMICFG1 | ECM n maskable interrupt configuration register 1 | √ | √ | √ | √ | — |
| ECMnMICFG2 | ECM n maskable interrupt configuration register 2 | √ | √ | √ | √ | — |
| ECMnNMICFG0 | ECM n non-maskable interrupt configuration register 0 | √ | √ | √ | √ | — |
| ECMnNMICFG1 | ECM n non-maskable interrupt configuration register 1 | √ | √ | √ | √ | — |
| ECMnNMICFG2 | ECM n non-maskable interrupt configuration register 2 | √ | √ | √ | √ | — |
| ECMnIRCFG0 | ECM n internal reset configuration register 0 | √ | √ | √ | √ | — |
| ECMnIRCFG1 | ECM n internal reset configuration register 1 | √ | √ | √ | √ | — |
| ECMnIRCFG2 | ECM n internal reset configuration register 2 | √ | √ | √ | √ | — |
| ECMnEMK0 | ECM n error mask register 0 | √ | √ | √ | √ | — |

**Table 29.13 Reset condition of ECM Common Registers (2/2)**

| Register Symbol | Register Name | Reset condition | | | | |
|---|---|---|---|---|---|---|
| | | Power On Reset | System Reset1 | System Reset2 | Application Reset1 | Limited Reset |
| ECMnEMK1 | ECM n error mask register 1 | √ | √ | √ | √ | — |
| ECMnEMK2 | ECM n error mask register 2 | √ | √ | √ | √ | — |
| ECMnESSTC0 | ECM n error source status clear register 0 | √ | √ | √ | √ | — |
| ECMnESSTC1 | ECM n error source status clear register 1 | √ | √ | √ | √ | — |
| ECMnESSTC2 | ECM n error source status clear register 2 | √ | √ | √ | √ | — |
| ECMnPCMD1 | ECM n protection command register | √ | √ | √ | √ | — |
| ECMnPS | ECM n protection status register | √ | √ | √ | √ | — |
| ECMnPE0 | ECM n pseudo error trigger register 0 | √ | √ | √ | √ | — |
| ECMnPE1 | ECM n pseudo error trigger register 1 | √ | √ | √ | √ | — |
| ECMnPE2 | ECM n pseudo error trigger register 2 | √ | √ | √ | √ | — |
| ECMnDTMCTL | ECM n delay timer control register | √ | √ | √ | √ | — |
| ECMnDTMR | ECM n delay timer register | √ | √ | √ | √ | — |
| ECMnDTMCMP | ECM n delay timer compare register | √ | √ | √ | √ | — |
| ECMnDTMCFG0 | ECM n delay timer configuration register 0 | √ | √ | √ | √ | — |
| ECMnDTMCFG1 | ECM n delay timer configuration register 1 | √ | √ | √ | √ | — |
| ECMnDTMCFG2 | ECM n delay timer configuration register 2 | √ | √ | √ | √ | — |
| ECMnDTMCFG3 | ECM n delay timer configuration register 3 | √ | √ | √ | √ | — |
| ECMnDTMCFG4 | ECM n delay timer configuration register 4 | √ | √ | √ | √ | — |
| ECMnDTMCFG5 | ECM n delay timer configuration register 5 | √ | √ | √ | √ | — |
| ECMnEOCCFG | ECM n error output clear invalidation configuration register | √ | √ | √ | — | — |
| ECMnPEM | ECM n pseudo error mask register | √ | √ | √ | √ | — |

## 29.4.2 ECMmnESET — ECM Master/Checker n Error Set Trigger Register (m = M/C)

The ECM master/checker n error set trigger register is for setting the error signal from the error pin to the low level. When the ECMmEST bit is set to 1, the error pin immediately outputs the low level. The output cannot be masked. You have to follow a predetermined sequence for writing data to this register. See **Section 29.3.5, Writing to Protected Registers**, for the details of the write protection sequence. This register maintains "1" only 1 PCLK cycle.

This register is always read as 0000 $0000_H$.

**Access:** This register can be written in 32-bit units.

**Address:** <ECMMn_base>
<ECMCn_base>

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ECMmEST |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 29.14 ECMmnESET register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | The write value must always be 0. |
| 0 | ECMmEST | Error set trigger bit<br>0: Writing 0 is invalid<br>1: Set the output level from the error pin to the active (low) level. |

**CAUTIONS**

Setting or clearing the error output from the error pin via the ECMmnESET or ECMmnECLR register will set the ECMmSSE228 bit of the ECMmnESSTR2 register (ECM compare error). Therefore, the ECMmnESET register has to be set following the sequence below.

1. Set the MSKM bit and MSKC bit of the ECMnPEM register to "masked".
2. Set the ECMmEST bit in the ECMmnESET register.
3. Wait until ERROROUTZ becomes low by checking that the ECMmSSE231 bit of the ECMmnESSTR2 register is "0" .
4. Set the MSKM bit and MSKC bit of the ECMnPEM register to "not masked".

### 29.4.3 ECMmnECLR — ECM Master/Checker n Error Clear Trigger Register (m = M/C)

The ECM master/checker n error clear trigger register is for setting the error signal from the error pin to the high level (toggle). When the ECMmECT bit is set to 1, the error pin outputs the high level (toggle) as long as there are no other sources that set the error pin to the low level. You have to follow a predetermined sequence for writing data to this register. See **Section 29.3.5, Writing to Protected Registers**, for the details of the write protection sequence. This register is always read as $0000\ 0000_H$.

**Access:** This register can be written in 32-bit units.

**Address:** <ECMMn_base> + $04_H$
<ECMCn_base> + $04_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ECMmECT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

**Table 29.15 ECMmnECLR register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | The write value must always be 0. |
| 0 | ECMmECT | Error clear trigger bit<br>0: Writing 0 is invalid<br>1: Set the output level from the error pin to the inactive (high) level. |

**CAUTIONS**

Clearing of the error pin is only possible if all errors, not masked by ECMnEMK0/1/2, are cleared beforehand.

Setting or clearing the error output via the ECMmnECLR register will generate the error. Therefore, the following has to be set in advance. The sequence below shall be executed by either CPU.

1. Set the MSKM bit and MSKC bit of the ECMnPEM register to “masked”.
2. Set the ECMmECT bit in the ECMmnECLR registers.
3. Wait until ERROROUTZ becomes high by reading ECMmSSE231 bit of ECMm0ESSTR2 or ECMm1ESSTR2 register 30 times. After that, check that the ECMmSSE231 bit of the ECMmnESSTR2 register ”is “1”. If the ECMmSSE231 bit of the ECMmnESSTR2 register is not “1”, a new error may occur.
4. Set the MSKM bit and MSKC bit of the ECM0nPEM register to “not masked”.

Note: This procedure is in case of not setting ECMnEOCCFG register.

Setting both ECMmnECLR (n = 0,1) registers are required even PE2 is disabled on P1H-C (4MB), P1H-C (8MB) and P1H-CE except P1M-C/P1M-C ED mode.

**NOTE**

If the ErrorPin Low Time counter is still running, the Error Output clear function is masked. The Error Output clear request will not be memorized. Error Output clear function is executed, if the ErrorPin Low Time Counter is expired and the Error Output clear register is written.

### 29.4.4 ECMmnESSTR0 — ECM Master/Checker n Error Source Status Register 0 (m = M/C)

The ECM master/checker n error source status register 0 is a read-only register.

This register represents the status of individual internal error sources, which is irrelevant to the setting of the error mask. The status can be cleared only by software and power on reset. Other reset will not affect the status.

**Access:** This register can be read in 32-bit units.

**Address:** <ECMMn_base> + $08_H$
<ECMCn_base> + $08_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMm SSE031 | ECMm SSE030 | ECMm SSE029 | ECMm SSE028 | ECMm SSE027 | ECMm SSE026 | ECMm SSE025 | ECMm SSE024 | ECMm SSE023 | ECMm SSE022 | ECMm SSE021 | ECMm SSE020 | ECMm SSE019 | ECMm SSE018 | ECMm SSE017 | ECMm SSE016 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMm SSE015 | ECMm SSE014 | ECMm SSE013 | ECMm SSE012 | ECMm SSE011 | ECMm SSE010 | ECMm SSE009 | ECMm SSE008 | ECMm SSE007 | ECMm SSE006 | ECMm SSE005 | ECMm SSE004 | ECMm SSE003 | ECMm SSE002 | ECMm SSE001 | ECMm SSE000 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 29.16 ECMmnESSTR0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMmSSE031 to ECMmSSE000 | Error source status bit<br>ECMmSSE031 to ECMmSSE000 correspond to error sources 31 to 0.<br>0: Error not occurred<br>1: Error occurred |

### 29.4.5 ECMmnESSTR1 — ECM Master/Checker n Error Source Status Register 1 (m = M/C)

The ECM master/checker error source status register 1 is a read-only register.

This register represents the status of individual internal error sources, which is irrelevant to the setting of the error mask. The status can be cleared only by software and power on reset. Other reset will not affect the status.

**Access:** This register can be read in 32-bit units.

**Address:** <ECMMn_base> + $0C_H$
<ECMCn_base> + $0C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMm SSE131 | ECMm SSE130 | ECMm SSE129 | ECMm SSE128 | ECMm SSE127 | ECMm SSE126 | ECMm SSE125 | ECMm SSE124 | ECMm SSE123 | ECMm SSE122 | ECMm SSE121 | ECMm SSE120 | ECMm SSE119 | ECMm SSE118 | ECMm SSE117 | ECMm SSE116 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMm SSE115 | ECMm SSE114 | ECMm SSE113 | ECMm SSE112 | ECMm SSE111 | ECMm SSE110 | ECMm SSE109 | ECMm SSE108 | ECMm SSE107 | ECMm SSE106 | ECMm SSE105 | ECMm SSE104 | ECMm SSE103 | ECMm SSE102 | ECMm SSE101 | ECMm SSE100 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 29.17 ECMmnESSTR1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMmSSE131 to ECMmSSE100 | Error source status bit<br>ECMmSSE131 to ECMmSSE100 correspond to error sources 63 to 32.<br>0: Error not occurred<br>1: Error occurred |

### 29.4.6 ECMmnESSTR2 — ECM Master/Checker n Error Source Status Register 2 (m = M/C)

The ECM master/checker n error source status register 2 is a read-only register.

This register represents the status of individual internal error sources, which is irrelevant to the setting of the error mask. The status can be cleared only by software and power on reset. Other reset will not affect the status.

**Access:** This register can be read in 32-bit units.

**Address:** <ECMMn_base> + $10_H$
<ECMCn_base> + $10_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMm SSE231 | ECMm SSE230 | ECMm SSE229 | ECMm SSE228 | ECMm SSE227 | ECMm SSE226 | ECMm SSE225 | ECMm SSE224 | ECMm SSE223 | ECMm SSE222 | ECMm SSE221 | ECMm SSE220 | ECMm SSE219 | ECMm SSE218 | ECMm SSE217 | ECMm SSE216 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMm SSE215 | ECMm SSE214 | ECMm SSE213 | ECMm SSE212 | ECMm SSE211 | ECMm SSE210 | ECMm SSE209 | ECMm SSE208 | ECMm SSE207 | ECMm SSE206 | ECMm SSE205 | ECMm SSE204 | ECMm SSE203 | ECMm SSE202 | ECMm SSE201 | ECMm SSE200 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 29.18 ECMmnESSTR2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | ECMmSSE231 | The status of the ERROROUTZ pin<br>0: ERROROUTZ is low level<br>1: ERROROUTZ is high level |
| 30 | ECMmSSE230 | The status of the ECMmnESET writing<br>0: No Error<br>1: Error is set by ECMmnESET |
| 29 | ECMmSSE229 | The status of the delay timer overflow<br>0: No overflow<br>1: Overflow |
| 28 to 0 | ECMmSSE228 to ECMmSSE200 | Error source status bit<br>ECMmSSE228 to ECMmSSE200 correspond to error sources 92 to 64.<br>0: Error not occurred<br>1: Error occurred |

**NOTE**

After Field-BIST executes whether it is unintended execution or not, ECMmnESSTR2.bit 27 will be always “1”. In the startup (after reset release), Field-BIST will be always executed and “1” will be captured in ECMmnESSTR2.bit27.

Therefore confirm ECMmnESSTR2.bit27 is “1” and clear ECMmnESSTR2.bit27 by software before user operation runs.

### 29.4.7 ECMmnPCMD0 — ECM Master/Checker n Protection Command Register (m = M/C)

The ECM master/checker n protection command register is a write-only register and can be written in 32-bit units. See **Section 29.4.1, List of Registers**, for the protected registers.

See **Section 29.3.5, Writing to Protected Registers**, for the details of the write protection sequence. The value after reset is undefined.

**Access:** This register can be written in 32-bit units.

**Address:** <ECMMn_base> + $14_H$
<ECMCn_base> + $14_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ECMm0REG[7:0] | | | | | | | |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 29.19 ECMmnPCMD0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | The write value must always be 0. |
| 7 to 0 | ECMmREG0 [7:0] | Protection command that enables writing to write protected ECMm registers. |

### 29.4.8 ECMnEPCFG — ECM n Error Pulse Configuration Register

The ECM n error pulse configuration register is a read/write register. Writing to this register is protected by a sequence of instructions.

For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base>

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ECMSL0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 29.20 ECMnEPCFG register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | |
| 0 | ECMSL0 | Error pin operation configuration bit<br>Error output operation setting for the error pin<br>0: Non-dynamic mode<br>1: Dynamic mode |

In Dynamic mode the timer output GTMAT0O5 determines the output wave of the error terminal in case of no error.

**CAUTION**

After setting the dynamic mode, it is recommended not to change to non-dynamic mode again, because there is a possibility of a glitch at the error output.

### 29.4.9 ECMnMICFG0 — ECM n Maskable Interrupt Configuration Register 0

The ECM n maskable interrupt configuration register 0 is used to set the generation of the INTECMnMI interrupts (EI level interrupts). The generation of EI level interrupts in response to errors is selectable. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $04_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMMIE031 | ECMMIE030 | ECMMIE029 | ECMMIE028 | ECMMIE027 | ECMMIE026 | ECMMIE025 | ECMMIE024 | ECMMIE023 | ECMMIE022 | ECMMIE021 | ECMMIE020 | ECMMIE019 | ECMMIE018 | ECMMIE017 | ECMMIE016 |
| Value after rest | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMMIE015 | ECMMIE014 | ECMMIE013 | ECMMIE012 | ECMMIE011 | ECMMIE010 | ECMMIE009 | ECMMIE008 | ECMMIE007 | ECMMIE006 | ECMMIE005 | ECMMIE004 | ECMMIE003 | ECMMIE002 | ECMMIE001 | ECMMIE000 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.21 ECMnMICFG0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMMIE031 to ECMMIE000 | ECM maskable interrupt generation control bit<br>ECMMIE031 to ECMMIE000 correspond to error sources 31 to 0.<br>0: Interrupt generation disabled<br>1: Interrupt generation enabled |

## 29.4.10 ECMnMICFG1 — ECM n Maskable Interrupt Configuration Register 1

The ECM n maskable interrupt configuration register 1 is used to set the generation of the INTECMnMI interrupts (EI level interrupts). The generation of EI level interrupts in response to errors is selectable. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $08_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMMIE131 | ECMMIE130 | ECMMIE129 | ECMMIE128 | ECMMIE127 | ECMMIE126 | ECMMIE125 | ECMMIE124 | ECMMIE123 | ECMMIE122 | ECMMIE121 | ECMMIE120 | ECMMIE119 | ECMMIE118 | ECMMIE117 | ECMMIE116 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMMIE115 | ECMMIE114 | ECMMIE113 | ECMMIE112 | ECMMIE111 | ECMMIE110 | ECMMIE109 | ECMMIE108 | ECMMIE107 | ECMMIE106 | ECMMIE105 | ECMMIE104 | ECMMIE103 | ECMMIE102 | ECMMIE101 | ECMMIE100 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.22 ECMnMICFG1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMMIE131 to ECMMIE100 | ECM maskable interrupt generation control bit<br>ECMMIE131 to ECMMIE100 correspond to error sources 63 to 32.<br>0: Interrupt generation disabled<br>1: Interrupt generation enabled |

### 29.4.11 ECMnMICFG2 — ECM n Maskable Interrupt Configuration Register 2

The ECM n maskable interrupt configuration register 2 is used to set the generation of INTECMnMI interrupts (EI level interrupts). The generation of EI level interrupts in response to errors is selectable. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $0C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ECMMIE228 | ECMMIE227 | ECMMIE226 | ECMMIE225 | ECMMIE224 | ECMMIE223 | ECMMIE222 | ECMMIE221 | ECMMIE220 | ECMMIE219 | ECMMIE218 | ECMMIE217 | ECMMIE216 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMMIE215 | ECMMIE214 | ECMMIE213 | ECMMIE212 | ECMMIE211 | ECMMIE210 | ECMMIE209 | ECMMIE208 | ECMMIE207 | ECMMIE206 | ECMMIE205 | ECMMIE204 | ECMMIE203 | ECMMIE202 | ECMMIE201 | ECMMIE200 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.23 ECMnMICFG2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | The write value must always be 0. |
| 28 to 0 | ECMMIE228 to ECMMIE200 | ECM maskable interrupt generation control bit<br>ECMMIE228 to ECMMIE200 correspond to error sources 92 to 64.<br>0: Interrupt generation disabled<br>1: Interrupt generation enabled |

## 29.4.12 ECMnNMICFG0 — ECM n Non-maskable Interrupt Configuration Register 0

The ECM n Non-maskable interrupt configuration register 0 is used to set the generation of INTECMnNMI interrupts (FE level interrupt). Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units

**Address:** <ECMn_base> + $10_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMN MIE031 | ECMN MIE030 | ECMN MIE029 | ECMN MIE028 | ECMN MIE027 | ECMN MIE026 | ECMN MIE025 | ECMN MIE024 | ECMN MIE023 | ECMN MIE022 | ECMN MIE021 | ECMN MIE020 | ECMN MIE019 | ECMN MIE018 | ECMN MIE017 | ECMN MIE016 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMN MIE015 | ECMN MIE014 | ECMN MIE013 | ECMN MIE012 | ECMN MIE011 | ECMN MIE010 | ECMN MIE009 | ECMN MIE008 | ECMN MIE007 | ECMN MIE006 | ECMN MIE005 | ECMN MIE004 | ECMN MIE003 | ECMN MIE002 | ECMN MIE001 | ECMN MIE000 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.24 ECMnNMICFG0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMNMIE031 to ECMNMIE000 | ECM Non-maskable interrupt generation control bit<br>ECMNMIE031 to ECMNMIE000 correspond to error sources 31 to 0.<br>0: Interrupt generation disabled<br>1: Interrupt generation enabled |

### 29.4.13 ECMnNMICFG1 — ECM n Non-maskable Interrupt Configuration Register 1

The ECM n Non-maskable level interrupt configuration register 1 is used to set the generation of INTECMnNMI interrupts (FE level interrupt). Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $14_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMN MIE131 | ECMN MIE130 | ECMN MIE129 | ECMN MIE128 | ECMN MIE127 | ECMN MIE126 | ECMN MIE125 | ECMN MIE124 | ECMN MIE123 | ECMN MIE122 | ECMN MIE121 | ECMN MIE120 | ECMN MIE119 | ECMN MIE118 | ECMN MIE117 | ECMN MIE116 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMN MIE115 | ECMN MIE114 | ECMN MIE113 | ECMN MIE112 | ECMN MIE111 | ECMN MIE110 | ECMN MIE109 | ECMN MIE108 | ECMN MIE107 | ECMN MIE106 | ECMN MIE105 | ECMN MIE104 | ECMN MIE103 | ECMN MIE102 | ECMN MIE101 | ECMN MIE100 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.25 ECMnNMICFG1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMNMIE131 to ECMNMIE100 | ECM Non-maskable interrupt generation control bit<br>ECMNMIE131 to ECMNMIE100 correspond to error sources 63 to 32.<br>0: Interrupt generation disabled<br>1: Interrupt generation enabled |

### 29.4.14 ECMnNMICFG2 — ECM n Non-maskable Interrupt Configuration Register 2

The ECM n Non-maskable interrupt configuration register 2 is used to set the generation of INTECMnNMI interrupts (FE level interrupt). Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $18_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ECMNMIE228 | ECMNMIE227 | ECMNMIE226 | ECMNMIE225 | ECMNMIE224 | ECMNMIE223 | ECMNMIE222 | ECMNMIE221 | ECMNMIE220 | ECMNMIE219 | ECMNMIE218 | ECMNMIE217 | ECMNMIE216 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMNMIE215 | ECMNMIE214 | ECMNMIE213 | ECMNMIE212 | ECMNMIE211 | ECMNMIE210 | ECMNMIE209 | ECMNMIE208 | ECMNMIE207 | ECMNMIE206 | ECMNMIE205 | ECMNMIE204 | ECMNMIE203 | ECMNMIE202 | ECMNMIE201 | ECMNMIE200 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.26 ECMnNMICFG2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | The write value must always be 0. |
| 28 to 0 | ECMNMIE228 to ECMNMIE200 | ECM Non-maskable interrupt generation control bit<br>ECMNMIE228 to ECMNMIE200 correspond to error sources 92 to 64.<br>0: Interrupt generation disabled<br>1: Interrupt generation enabled |

## 29.4.15 ECMnIRCFG0 — ECM n Internal Reset Configuration Register 0

The ECM n internal reset configuration register 0 is used to set the generation of Error Control Module Reset in response to internal errors. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $1C_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMIRE031 | ECMIRE030 | ECMIRE029 | ECMIRE028 | ECMIRE027 | ECMIRE026 | ECMIRE025 | ECMIRE024 | ECMIRE023 | ECMIRE022 | ECMIRE021 | ECMIRE020 | ECMIRE019 | ECMIRE018 | ECMIRE017 | ECMIRE016 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMIRE015 | ECMIRE014 | ECMIRE013 | ECMIRE012 | ECMIRE011 | ECMIRE010 | ECMIRE009 | ECMIRE008 | ECMIRE007 | ECMIRE006 | ECMIRE005 | ECMIRE004 | ECMIRE003 | ECMIRE002 | ECMIRE001 | ECMIRE000 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.27 ECMnIRCFG0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMIRE031 to ECMIRE000 | ECM internal reset generation control bit<br>ECMIRE031 to ECMIRE000 correspond to error sources 31 to 0.<br>0: Error Control Module Reset generation disabled<br>1: Error Control Module Reset generation enabled |

**NOTE**

Only watch dog timer error, default setting of Error Control Module Reset configuration is “Enabled”. See **Table 29.8, List of Error Inputs.** This is mandatory to support the automatic start mode of the WDTA. The start mode is selectable by flash option setting. The WDTA provides two modes for the counter start after reset release:

- Software trigger start mode: The counter value remains $0000_H$ after reset release. The counter is started with the first WDTA trigger. The first trigger can occur any time after reset release.
- Automatic start mode: The counter starts automatically after reset release. The first trigger must occur before the counter overflows.

### 29.4.16 ECMnIRCFG1 — ECM n Internal Reset Configuration Register 1

The ECM n internal reset configuration register 1 is used to set the generation of Error Control Module Reset in response to internal errors. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $20_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMIRE131 | ECMIRE130 | ECMIRE129 | ECMIRE128 | ECMIRE127 | ECMIRE126 | ECMIRE125 | ECMIRE124 | ECMIRE123 | ECMIRE122 | ECMIRE121 | ECMIRE120 | ECMIRE119 | ECMIRE118 | ECMIRE117 | ECMIRE116 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMIRE115 | ECMIRE114 | ECMIRE113 | ECMIRE112 | ECMIRE111 | ECMIRE110 | ECMIRE109 | ECMIRE108 | ECMIRE107 | ECMIRE106 | ECMIRE105 | ECMIRE104 | ECMIRE103 | ECMIRE102 | ECMIRE101 | ECMIRE100 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.28 ECMnIRCFG1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMIRE131 to ECMIRE100 | ECM internal reset generation control bit<br>ECMIRE131 to ECMIRE100 correspond to error sources 63 to 32.<br>0: Error Control Module Reset generation disabled<br>1: Error Control Module Reset generation enabled |

## 29.4.17 ECMnIRCFG2 — ECM n Internal Reset Configuration Register 2

The ECM n internal reset configuration register 2 is used to set the generation of Error Control Module Reset in response to internal errors. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $24_H$

**Value after reset:** $00000000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | ECMIRE229 | ECMIRE228 | ECMIRE227 | ECMIRE226 | ECMIRE225 | ECMIRE224 | ECMIRE223 | ECMIRE222 | ECMIRE221 | ECMIRE220 | ECMIRE219 | ECMIRE218 | ECMIRE217 | ECMIRE216 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMIRE215 | ECMIRE214 | ECMIRE213 | ECMIRE212 | ECMIRE211 | ECMIRE210 | ECMIRE209 | ECMIRE208 | ECMIRE207 | ECMIRE206 | ECMIRE205 | ECMIRE204 | ECMIRE203 | ECMIRE202 | ECMIRE201 | ECMIRE200 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.29 ECMnIRCFG2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | The write value must always be 0. |
| 29 | ECMIRE229 | ECM internal reset control bit. Corresponds to delay timer overflow.<br>0: Error Control Module Reset generation disabled.<br>1: Error Control Module Reset generation enabled. |
| 28 to 0 | ECMIRE228 to ECMIRE200 | ECM internal reset generation control bit<br>ECMIRE228 to ECMIRE200 correspond to error sources 92 to 64.<br>0: Error Control Module Reset generation disabled<br>1: Error Control Module Reset generation enabled |

## 29.4.18 ECMnEMK0 — ECM n Error Mask Register 0

The ECM n error mask register 0 is used to mask the individual error sources of the error pin output. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $28_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECME MK031 | ECME MK030 | ECME MK029 | ECME MK028 | ECME MK027 | ECME MK026 | ECME MK025 | ECME MK024 | ECME MK023 | ECME MK022 | ECME MK021 | ECME MK020 | ECME MK019 | ECME MK018 | ECME MK017 | ECME MK016 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECME MK015 | ECME MK014 | ECME MK013 | ECME MK012 | ECME MK011 | ECME MK010 | ECME MK009 | ECME MK008 | ECME MK007 | ECME MK006 | ECME MK005 | ECME MK004 | ECME MK003 | ECME MK002 | ECME MK001 | ECME MK000 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.30 ECMnEMK0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMEMK031 to ECMEMK000 | ECM error output signal mask control bit<br>ECMEMK031 to ECMEMK000 correspond to error sources 31 to 0.<br>0: Error signal output not masked<br>1: Error signal output masked |

**NOTE**

If a error flag is set but masked, clearing the mask will set the ERROROUTZ to active level. This happens independently from the time of the error occurrence. In other words, the flag is evaluated statically.

When PE2 is disabled within P1H-C (4MB), P1H-C (8MB) and P1H-CE (except P1M-C/P1M-C ED mode), all error of ECM1 can be masked by ECM1EMK0/1/2 to prevent unexpected error output.

## 29.4.19 ECMnEMK1 — ECM n Error Mask Register 1

The ECM n error mask register 1 is used to mask the individual error sources of the error pin output. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $2C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECME MK131 | ECME MK130 | ECME MK129 | ECME MK128 | ECME MK127 | ECME MK126 | ECME MK125 | ECME MK124 | ECME MK123 | ECME MK122 | ECME MK121 | ECME MK120 | ECME MK119 | ECME MK118 | ECME MK117 | ECME MK116 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECME MK115 | ECME MK114 | ECME MK113 | ECME MK112 | ECME MK111 | ECME MK110 | ECME MK109 | ECME MK108 | ECME MK107 | ECME MK106 | ECME MK105 | ECME MK104 | ECME MK103 | ECME MK102 | ECME MK101 | ECME MK100 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.31 ECMnEMK1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMEMK131 to ECMEMK100 | ECM error output signal mask control bit<br>ECMEMK131 to ECMEMK100 correspond to error sources 63 to 32.<br>0: Error signal output not masked<br>1: Error signal output masked |

**NOTE**

If a error flag is set but masked, clearing the mask will set the ERROROUTZ to active level. This happens independently from the time of the error occurrence. In other words, the flag is evaluated statically.

When PE2 is disabled within P1H-C (4MB), P1H-C (8MB) and P1H-CE (except P1M-C/P1M-C ED mode), all error of ECM1 can be masked by ECM1EMK0/1/2 to prevent unexpected error output.

## 29.4.20 ECMnEMK2 — ECM n Error Mask Register 2

The ECM n error mask register 2 is used to mask the individual error sources of the error pin output. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $30_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | ECMEMK229 | ECMEMK228 | ECMEMK227 | ECMEMK226 | ECMEMK225 | ECMEMK224 | ECMEMK223 | ECMEMK222 | ECMEMK221 | ECMEMK220 | ECMEMK219 | ECMEMK218 | ECMEMK217 | ECMEMK216 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMEMK215 | ECMEMK214 | ECMEMK213 | ECMEMK212 | ECMEMK211 | ECMEMK210 | ECMEMK209 | ECMEMK208 | ECMEMK207 | ECMEMK206 | ECMEMK205 | ECMEMK204 | ECMEMK203 | ECMEMK202 | ECMEMK201 | ECMEMK200 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.32 ECMnEMK2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | The write value must always be 0. |
| 29 | ECMEMK229 | Corresponds to delay timer overflow.<br>0: Error signal output is not masked.<br>1: Error signal output is masked. |
| 28 to 0 | ECMEMK228 to ECMEMK200 | ECM error output signal mask control bit<br>ECMEMK228 to ECMEMK200 correspond to error sources 92 to 64.<br>0: Error signal output not masked<br>1: Error signal output masked |

**NOTE**

If a error flag is set but masked, clearing the mask will set the ERROROUTZ to active level. This happens independently from the time of the error occurrence. In other words, the flag is evaluated statically.

When PE2 is disabled within P1H-C (4MB), P1H-C (8MB) and P1H-CE (except P1M-C/P1M-C ED mode), all error of ECM1 can be masked by ECM1EMK0/1/2 to prevent unexpected error output.

## 29.4.21 ECMnESSTC0 — ECM n Error Source Status Clear Trigger Register 0

The ECM n error source status clear trigger register 0 is a write-only register and can be written in 32-bit units. This register is used to clear the individual error source status of the ECM master/checker error source status register 0. Both the error status of the ECM master and the ECM checker are cleared simultaneously.

Writing to this register is protected by the instruction sequence. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be written in 32-bit units.

**Address:** <ECMn_base> + $34_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMCLSSE031 | ECMCLSSE030 | ECMCLSSE029 | ECMCLSSE028 | ECMCLSSE027 | ECMCLSSE026 | ECMCLSSE025 | ECMCLSSE024 | ECMCLSSE023 | ECMCLSSE022 | ECMCLSSE021 | ECMCLSSE020 | ECMCLSSE019 | ECMCLSSE018 | ECMCLSSE017 | ECMCLSSE016 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMCLSSE015 | ECMCLSSE014 | ECMCLSSE013 | ECMCLSSE012 | ECMCLSSE011 | ECMCLSSE010 | ECMCLSSE009 | ECMCLSSE008 | ECMCLSSE007 | ECMCLSSE006 | ECMCLSSE005 | ECMCLSSE004 | ECMCLSSE003 | ECMCLSSE002 | ECMCLSSE001 | ECMCLSSE000 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 29.33 ECMnESSTC0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMCLSSE031 to ECMCLSSE000 | ECM error status clear bit<br>ECMCLSSE031 to ECMCLSSE000 correspond to ECMmSSE031 to ECMmSSE000.<br>0: Corresponding error status unchanged<br>1: Corresponding error status cleared |

## 29.4.22 ECMnESSTC1 — ECM n Error Source Status Clear Trigger Register 1

The ECM n error source status clear trigger register 1 is a write-only register and can be written in 32-bit units. This register is used to clear the individual error source status of the ECM master/checker error source status register 1. Both the error status of the ECM master and the ECM checker are cleared simultaneously.

Writing to this register is protected by the instruction sequence. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be written in 32-bit units.

**Address:** <ECMn_base> + $38_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMCLSSE131 | ECMCLSSE130 | ECMCLSSE129 | ECMCLSSE128 | ECMCLSSE127 | ECMCLSSE126 | ECMCLSSE125 | ECMCLSSE124 | ECMCLSSE123 | ECMCLSSE122 | ECMCLSSE121 | ECMCLSSE120 | ECMCLSSE119 | ECMCLSSE118 | ECMCLSSE117 | ECMCLSSE116 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMCLSSE115 | ECMCLSSE114 | ECMCLSSE113 | ECMCLSSE112 | ECMCLSSE111 | ECMCLSSE110 | ECMCLSSE109 | ECMCLSSE108 | ECMCLSSE107 | ECMCLSSE106 | ECMCLSSE105 | ECMCLSSE104 | ECMCLSSE103 | ECMCLSSE102 | ECMCLSSE101 | ECMCLSSE100 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 29.34 ECMnESSTC1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMCLSSE131 to ECMCLSSE100 | ECM error status clear bit<br>ECMCLSSE131 to ECMCLSSE100 correspond to ECMmSSE131 to ECMmSSE100.<br>0: Corresponding error status unchanged<br>1: Corresponding error status cleared |

## 29.4.23 ECMnESSTC2 — ECM n Error Source Status Clear Trigger Register 2

The ECM n error source status clear trigger register 2 is a write-only register and can be written in 32-bit units. This register is used to clear the individual error source status of the ECM master/checker error source status register 2. Both the error status of the ECM master and the ECM checker are cleared simultaneously.

Writing to this register is protected by the instruction sequence. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be written in 32-bit units.

**Address:** <ECMn_base> + $3C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | ECMCLSSE230 | ECMCLSSE229 | ECMCLSSE228 | ECMCLSSE227 | ECMCLSSE226 | ECMCLSSE225 | ECMCLSSE224 | ECMCLSSE223 | ECMCLSSE222 | ECMCLSSE221 | ECMCLSSE220 | ECMCLSSE219 | ECMCLSSE218 | ECMCLSSE217 | ECMCLSSE216 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMCLSSE215 | ECMCLSSE214 | ECMCLSSE213 | ECMCLSSE212 | ECMCLSSE211 | ECMCLSSE210 | ECMCLSSE209 | ECMCLSSE208 | ECMCLSSE207 | ECMCLSSE206 | ECMCLSSE205 | ECMCLSSE204 | ECMCLSSE203 | ECMCLSSE202 | ECMCLSSE201 | ECMCLSSE200 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 29.35 ECMnESSTC2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | The write value must always be 0. |
| 30 | ECMCLSSE230 | ECM error status clear bit<br>ECMCLSSE230 corresponds to ECMmSSE230.<br>0: Error status unchanged<br>1: Error status cleared |
| 29 | ECMCLSSE229 | ECM error status clear bit<br>ECMCLSSE229 corresponds to ECMmSSE229.<br>0: Error status unchanged<br>1: Error status cleared |
| 28 to 0 | ECMCLSSE228 to ECMCLSSE200 | ECM error status clear bit<br>ECMCLSSE228 to ECMCLSSE200 correspond to ECMmSSE228 to ECMmSSE200.<br>0: Corresponding error status unchanged<br>1: Corresponding error status cleared |

## 29.4.24 ECMnPCMD1 — ECM n Protection Command Register

The ECM n protection command register is a write-only register and can be written in 32-bit units. See **Section 29.4.1, List of Registers**, for the protected registers. See **Section 29.3.5, Writing to Protected Registers**, for the details of the write protection sequence. The value after reset is undefined.

**Access:** This register can be written in 32-bit units.

**Address:** <ECMn_base> + $40_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ECMREG1[7:0] | | | | | | | |
| Value after reset | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 29.36 ECMnPCMD1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | The write value must always be 0. |
| 7 to 0 | ECMREG1[7:0] | Protection command that enables writing to write protected ECM registers. |

## 29.4.25 ECMnPS — ECM n Protection Status Register

The ECM n protection status register is a read-only register. This register is used to verify the write protected register has been written successfully or not. See **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read in 8-bit units.

**Address:** <ECMn_base> + $44_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ECMPRERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

**Table 29.37 ECMnPS register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 1 | Reserved | The write value must always be 0. |
| 0 | ECMPRERR | ECM protection status bit<br>Indicates whether writing to a write protected register was failed or was successful.<br>0: Writing was successfully completed.<br>1: Writing failed |

### 29.4.26 ECMnPE0 — ECM n Pseudo Error Trigger Register 0

The ECM n pseudo error trigger register 0 is a write-only register. This register is used to generate a pseudo error for test purposes. The ECM operation in response to the generation of a pseudo error is identical to that in response to a real error source. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be written in 32-bit units.

**Address:** <ECMn_base> + $48_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMPE031 | ECMPE030 | ECMPE029 | ECMPE028 | ECMPE027 | ECMPE026 | ECMPE025 | ECMPE024 | ECMPE023 | ECMPE022 | ECMPE021 | ECMPE020 | ECMPE019 | ECMPE018 | ECMPE017 | ECMPE016 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMPE015 | ECMPE014 | ECMPE013 | ECMPE012 | ECMPE011 | ECMPE010 | ECMPE009 | ECMPE008 | ECMPE007 | ECMPE006 | ECMPE005 | ECMPE004 | ECMPE003 | ECMPE002 | ECMPE001 | ECMPE000 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 29.38 ECMnPE0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMPE031 to ECMPE000 | ECM pseudo error trigger bit<br>ECMPE031 to ECMPE000 correspond to error sources 31 to 0.<br>0: Pseudo error is not generated.<br>1: Pseudo error is generated. |

## 29.4.27 ECMnPE1 — ECM n Pseudo Error Trigger Register 1

The ECM n pseudo error trigger register 1 is a write-only register. This register is used to generate a pseudo error for test purposes. The ECM operation in response to the generation of a pseudo error is identical to that in response to a real error source. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be written in 32-bit units.

**Address:** <ECMn_base> + $4C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMPE131 | ECMPE130 | ECMPE129 | ECMPE128 | ECMPE127 | ECMPE126 | ECMPE125 | ECMPE124 | ECMPE123 | ECMPE122 | ECMPE121 | ECMPE120 | ECMPE119 | ECMPE118 | ECMPE117 | ECMPE116 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMPE115 | ECMPE114 | ECMPE113 | ECMPE112 | ECMPE111 | ECMPE110 | ECMPE109 | ECMPE110 | ECMPE107 | ECMPE106 | ECMPE105 | ECMPE104 | ECMPE103 | ECMPE102 | ECMPE101 | ECMPE100 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 29.39 ECMnPE1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMnPE131 to ECMPE100 | ECM pseudo error trigger bit<br>ECMPE131 to ECMPE100 correspond to error sources 63 to 32.<br>0: Pseudo error is not generated.<br>1: Pseudo error is generated. |

## 29.4.28 ECMnPE2 — ECM n Pseudo Error Trigger Register 2

The ECM n pseudo error trigger register 2 is a write-only register. This register is used to generate a pseudo error for test purposes. The ECM operation in response to the generation of a pseudo error is identical to that in response to a real error source. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see Section Writing to Protected Registers.

**Access:** This register can be written in 32-bit units.

**Address:** <ECMn_base> + $50_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | ECM PE229 | ECM PE228 | — | ECM PE226 | ECM PE225 | ECM PE224 | ECM PE223 | ECM PE222 | ECM PE221 | ECM PE220 | ECM PE219 | ECM PE218 | ECM PE217 | ECM PE216 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | W | W | R | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECM PE215 | ECM PE214 | ECM PE213 | ECM PE212 | ECM PE211 | ECM PE210 | ECM PE209 | ECM PE208 | ECM PE207 | ECM PE206 | ECM PE205 | ECM PE204 | ECM PE203 | ECM PE202 | ECM PE201 | ECM PE200 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 29.40 ECMnPE2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | The write value must always be 0. |
| 29 | ECMPE229 | ECM pseudo error trigger bit. Corresponds to delay timer overflow.<br>0: Pseudo error is not generated.<br>1: Pseudo error is generated. |
| 28 | ECMPE228 | ECM pseudo error trigger bit. Corresponds to ECM compare error.<br>0: Pseudo error is not generated.<br>1: Pseudo error is generated. |
| 27 | Reserved | The write value must always be 0. |
| 26 to 0 | ECMPE226 to ECMPE200 | ECM pseudo error trigger bit<br>ECMPE226 to ECMPE200 correspond to error sources 90 to 64.<br>0: Pseudo error is not generated.<br>1: Pseudo error is generated. |

**NOTE**

After Field-BIST executes whether it is unintended execution or not, ECMmnESSTR2.bit 27 will be always “1” In the startup (after reset release), Field-BIST will be always executed and “1” will be captured in ECMmnESSTR2.bit27. If “1” is not captured in ECMmnESSTR2.bit27 after Field-BIST execution, then ECMmnESSTR2.bit 27 is out of order. Therefore, ECMnPE2.bit27 is reserved bit because it is possible to detect the failure of ECMmnESSTR2.bit27 without self-diagnosis by using ECMnPE2.

## 29.4.29 ECMnDTMCTL — ECM n Delay Timer Control Register

The ECM n delay timer control register is a read/write register. This register is used to control the delay timer. Writing to this register is protected by the instruction sequence. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $54_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | DTMSTACNTCLK | — | — | DTMSTP | DTMSTA |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W | R/W |

**Table 29.41 ECMnDTMCTL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | The write value must always be 0. |
| 4 | DTMSTACNTCLK | Delay timer start confirmation status.<br>0: Delay timer does not start.<br>1: Delay timer starts. |
| 3 to 2 | - | Reserved. The write value must always be 0 |
| 1 | DTMSTP | Delay timer stop bit<br>By writing “1” to this bit, delay timer is stopped (0 write is ignored).<br>Simultaneously, DTMSTA bit will be 0.<br>0: Delay timer is completed or not executed.<br>1: Stop request for delay timer is on execution. |
| 0 | DTMSTA | Delay timer start bit<br>Specifies the operation of the delay timer when any error event is occurred.<br>0: Delay timer does not start<br>1: Delay timer starts |

**NOTES**

1. ECMnDTMCTL register can be accessed via P-Bus but delay timer runs with not P-Bus clock butdedicated counter clock.
   Therefore, time rag exists between writing of ECMnDTMCTL and running of delay timer.
   DTMSTACNTCLK can be used to confirm whether delay timer is enabled or not. Please confirm again whether DTMSTA is updated or not after your write action.
2. ECMnDTMCTL register can be written only when (DTMSTA, DTMSTACNTCLK) = (0, 0) or (1,1)
   Please confirm the combination of DTMSTA and DTMSTACNTCLK before your write action.

## 29.4.30 ECMnDTMR — ECM n Delay Timer Register

The ECM n delay timer register is a read-only register. The ECM n delay timer register is initialized by setting the DTMSTA bit of the ECM n delay timer control register from 1 (timer in operation) to 0 (timer stops).

**Access:** This register can be read in 16-bit units.

**Address:** <ECMn_base> + $58_H$

**Value after reset:** $0000_H$

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMDTMR[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

## 29.4.31 ECMnDTMCMP — ECM n Delay Timer Compare Register

The ECM n delay timer compare register is a read/write register. The ECMmSSE229 bit is set when this register matches with the value of the ECM n delay timer register. Writing data to this register has to be conducted while the delay timer is stopped. Writing to this register is protected by the instruction sequence. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $5C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CMPW |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMDTMCMP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.42 ECMnDTMCMP register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | The write value must always be 0. |
| 16 | CMPW | Indicates on execution of ECMnDTMCMP register setting to counter clock domain<br>0: Not executed<br>1: On execution of setting ECMnDTMCMP |
| 15 to 0 | ECMDTMCMP [15:0] | Delay timer compare value |

**NOTES**

1. ECMDTMCMP can be accessible via P-Bus. But delay timer is run with not P-Bus clock but dedicated counter clock. When ECMDTMCMP is configured, this value is moved to dedicated counter clock domain to be able to be used by delay timer. CMPW indicates the current status of ECMDTMCMP setting on dedicated counter clock domain.
2. While CMPW is “1”, writing of ECMDTMCMP is ignored. Please confirm CMPW = 0 before writing of ECMDTMCMP.

## 29.4.32 ECMnDTMCFG0 — ECM n Delay Timer Configuration Register 0

The ECM n delay timer configuration register 0 is used to set enable/disable of the delay timer start caused by EI level interrupts in response to errors. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $60_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMTE031 | ECMTE030 | ECMTE029 | ECMTE028 | ECMTE027 | ECMTE026 | ECMTE025 | ECMTE024 | ECMTE023 | ECMTE022 | ECMTE021 | ECMTE020 | ECMTE019 | ECMTE018 | ECMTE017 | ECMTE016 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMTE015 | ECMTE014 | ECMTE013 | ECMTE012 | ECMTE011 | ECMTE010 | ECMTE009 | ECMTE008 | ECMTE007 | ECMTE006 | ECMTE005 | ECMTE004 | ECMTE003 | ECMTE002 | ECMTE001 | ECMTE000 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.43 ECMnDTMCFG0 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMTE031 to ECMTE000 | ECM delay timer start control bit<br>ECMTE031 to ECMTE000 correspond to EI level interrupts generated by error sources 31 to 0.<br>0: Delay timer start disabled<br>1: Delay timer start enabled |

**NOTE**

Do not enable delay timer for clock monitor error events (ECMTE012 to ECMTE008).

## 29.4.33 ECMnDTMCFG1 — ECM n Delay Timer Configuration Register 1

The ECM n delay timer configuration register 1 is used to set enable/disable of the delay timer start caused by EI level interrupts in response to errors. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $64_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMTE131 | ECMTE130 | ECMTE129 | ECMTE128 | ECMTE127 | ECMTE126 | ECMTE125 | ECMTE124 | ECMTE123 | ECMTE122 | ECMTE121 | ECMTE120 | ECMTE119 | ECMTE118 | ECMTE117 | ECMTE116 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMTE115 | ECMTE114 | ECMTE113 | ECMTE112 | ECMTE111 | ECMTE110 | ECMTE109 | ECMTE108 | ECMTE107 | ECMTE106 | ECMTE105 | ECMTE104 | ECMTE103 | ECMTE102 | ECMTE101 | ECMTE100 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.44 ECMnDTMCFG1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMTE131 to ECMTE100 | ECM delay timer start control bit<br>ECMTE131 to ECMTE100 correspond to EI level interrupts generated by error sources 63 to 32.<br>0: Delay timer start disabled<br>1: Delay timer start enabled |

## 29.4.34 ECMnDTMCFG2 — ECM n Delay Timer Configuration Register 2

The ECM n delay timer configuration register 2 is used to set enable/disable of the delay timer start caused by EI level interrupts in response to errors. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $68_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ECMTE228 | ECMTE227 | ECMTE226 | ECMTE225 | ECMTE224 | ECMTE223 | ECMTE222 | ECMTE221 | ECMTE220 | ECMTE219 | ECMTE218 | ECMTE217 | ECMTE216 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMTE215 | ECMTE214 | ECMTE213 | ECMTE212 | ECMTE211 | ECMTE210 | ECMTE209 | ECMTE208 | ECMTE207 | ECMTE206 | ECMTE205 | ECMTE204 | ECMTE203 | ECMTE202 | ECMTE201 | ECMTE200 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.45 ECMnDTMCFG2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | The write value must always be 0. |
| 28 to 0 | ECMTE228 to ECMTE200 | ECM delay timer start control bit<br>ECMTE228 to ECMTE200 correspond to EI level interrupts generated by error sources 92 to 64.<br>0: Delay timer start disabled<br>1: Delay timer start enabled |

## 29.4.35 ECMnDTMCFG3 — ECM n Delay Timer Configuration Register 3

The ECM delay timer configuration register 3 is a read/write register and can be written in 32-bit units. This register is used to set enable/disable of the delay timer start caused by FE level interrupts in response to errors. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $6C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMTE331 | ECMTE330 | ECMTE329 | ECMTE328 | ECMTE327 | ECMTE326 | ECMTE325 | ECMTE324 | ECMTE323 | ECMTE322 | ECMTE321 | ECMTE320 | ECMTE319 | ECMTE318 | ECMTE317 | ECMTE316 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMTE315 | ECMTE314 | ECMTE313 | ECMTE312 | ECMTE311 | ECMTE310 | ECMTE309 | ECMTE308 | ECMTE307 | ECMTE306 | ECMTE305 | ECMTE304 | ECMTE303 | ECMTE302 | ECMTE301 | ECMTE300 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.46 ECMnDTMCFG3 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMTE331 to ECMTE300 | ECM delay timer start control bit<br>ECMTE331 to ECMTE300 correspond to FE level interrupts generated by error sources 31 to 0.<br>0: Delay timer start disabled<br>1: Delay timer start enabled |

**NOTE**

Do not enable delay timer for clock monitor error events (ECMTE312 to ECMTE308).

## 29.4.36 ECMnDTMCFG4 — ECM n Delay Timer Configuration Register 4

The ECM n delay timer configuration register 4 is a read/write register and can be written in 32-bit units. This register is used to set enable/disable of the delay timer start caused by FE level interrupts in response to errors. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $70_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMTE431 | ECMTE430 | ECMTE429 | ECMTE428 | ECMTE427 | ECMTE426 | ECMTE425 | ECMTE424 | ECMTE423 | ECMTE422 | ECMTE421 | ECMTE420 | ECMTE419 | ECMTE418 | ECMTE417 | ECMTE416 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMTE415 | ECMTE414 | ECMTE413 | ECMTE412 | ECMTE411 | ECMTE410 | ECMTE409 | ECMTE408 | ECMTE407 | ECMTE406 | ECMTE405 | ECMTE404 | ECMTE403 | ECMTE402 | ECMTE401 | ECMTE400 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.47 ECMnDTMCFG4 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ECMTE431 to ECMTE400 | ECM delay timer start control bit<br>ECMTE431 to ECMTE400 correspond to FE level interrupts generated by error sources 63 to 32.<br>0: Delay timer start disabled<br>1: Delay timer start enabled |

## 29.4.37 ECMnDTMCFG5 — ECM n Delay Timer Configuration Register 5

The ECM n delay timer configuration register 5 is a read/write register and can be written in 32-bit units. This register is used to set enable/disable of the delay timer start caused by FE level interrupts in response to errors. Writing to this register is protected by a sequence of instructions. For details on the write protection sequence, see **Section 29.3.5, Writing to Protected Registers**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $74_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ECMTE528 | ECMTE527 | ECMTE526 | ECMTE525 | ECMTE524 | ECMTE523 | ECMTE522 | ECMTE521 | ECMTE520 | ECMTE519 | ECMTE518 | ECMTE517 | ECMTE516 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMTE515 | ECMTE514 | ECMTE513 | ECMTE512 | ECMTE511 | ECMTE510 | ECMTE509 | ECMTE508 | ECMTE507 | ECMTE506 | ECMTE505 | ECMTE504 | ECMTE503 | ECMTE502 | ECMTE501 | ECMTE500 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.48 ECMnDTMCFG5 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | The write value must always be 0. |
| 28 to 0 | ECMTE528 to ECMTE500 | ECM delay timer start control bit<br>ECMTE528 to ECMTE500 correspond to FE level interrupts generated by error sources 92 to 64.<br>0: Delay timer start disabled<br>1: Delay timer start enabled |

## 29.4.38 ECMnEOCCFG — ECM n Error Output Clear Invalidation Configuration Register

This register is readable/writable register. Access by 32-bit units is possible.

After counter for Error Output clear invalidation exceed the value which is configured to this register, it is possible to clear non-safe status of error output by SW.

Configure to this register only if error output status is safe.

This register initialized by a reset from Power on Reset or System Reset 1 or System Reset 2.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $78_H$

**Value after reset:** 0000 $0000_H$ This register is initialized by a reset from Power on Reset or System Reset 1 or System Reset 2.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CMPW |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECMEOUTCLRT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 29.49 ECMnEOCCFG register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 17 | Reserved | The write value must always be 0. |
| 16 | CMPW | Indicates on execution of ECMnEOCCFG register setting to counter clock domain<br>0: Not executed<br>1: On execution of setting ECMnEOCCFG |
| 15 to 0 | ECMEOUTCLRT[15:0] | The number of clock cycles after which it is possible to clear error output by SW. |

**NOTES**

1. ECMnEOCCFG can be accessible via P-Bus. But Errorout release timer is run with not P-Bus clock(PCLK) but dedicated counter clock (cntclk). When ECMnEOCCFG is configured, this value is moved to dedicated counter clock domain to be able to be used by Errorout release timer. CMPW indicates the current status of ECMnEOCCFG setting on dedicated counter clock domain.
2. While CMPW is “1”, writing of ECMnEOCCFG is ignored. Please confirm CMPW = 0 before writing of ECMnEOCCFG.

### 29.4.39 ECMnPEM — ECM n Pseudo error mask register

This register can mask the pseudo error of “ECM compare error” to support self-diagnosis of Errorout binder.

**Access:** This register can be read/written in 32-bit units.

**Address:** <ECMn_base> + $7C_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MSKM | MSKC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 29.50 ECMnPEM register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | The write value must always be 0. |
| 1 | MSKM | 0: Pseudo error of “ECM compare error” for ECM master is NOT masked.<br>1: Pseudo error of “ECM compare error” for ECM master is masked. |
| 0 | MSKC | 0: Pseudo error of “ECM compare error” for ECM checker is NOT masked.<br>1: Pseudo error of “ECM compare error” for ECM checker is masked. |

## 29.5 Difference among P1M-C, P1H-C and P1H-CE

- Number of ECM Units is different between P1M-C, P1H-C, P1H-CE shown in **Table 29.1**.
- Error inputs to ECM is different between P1M-C, P1H-C (4MB, BGA-292), P1H-C (4MB, BGA-156), P1H-C (8MB), P1H-CE shown in **Table 29.8**.

# Section 30 Data CRC Function B (DCRB)

This section contains a generic description of the Data CRC Function B (DCRB).

The first part of this section describes all RH850/P1x-C specific properties, such as the number of units, register base addresses, etc.

The remainder of the section describes the functions and registers of the DCRB.

## 30.1 Feature

### 30.1.1 Number of Units and Channels

This microcontroller has the following number of DCRB units.

Each DCRB unit has one channel DCRB.

**Table 30.1 Units and Channels**

| Product | P1M-C | P1H-C, P1H-CE |
|---|---|---|
| Number of Units | 4 | 8 |
| Name | DCRBn (n = 0 to 3) | DCRBn (n = 0 to 7) |

**Note:** Regarding the unit index “n” in this section, the individual DCRB units are identified by the index “n”; for example, DCRBnCTL indicates the DCRBn control register.

### 30.1.2 Register Base Address

DCRBn base addresses are listed in the following table.

DCRBn register addresses are given as offsets from the base addresses in general.

**Table 30.2 Register Base Address**

| | | Supporting Device | |
|---|---|---|---|
| Base Address Name | Base Address | P1M-C | P1H-C, P1H-CE |
| <DCRB0_base> | FFD5 $0000_H$ | √ | √ |
| <DCRB1_base> | FFF7 $0000_H$ | √ | √ |
| <DCRB2_base> | FFD5 $1000_H$ | √ | √ |
| <DCRB3_base> | FFF7 $1000_H$ | √ | √ |
| <DCRB4_base> | FFD5 $2000_H$ | — | √ |
| <DCRB5_base> | FFF7 $2000_H$ | — | √ |
| <DCRB6_base> | FFD5 $3000_H$ | — | √ |
| <DCRB7_base> | FFF7 $3000_H$ | — | √ |

### 30.1.3 Clock Supply

Clock supply by and to DCRBn is listed in the following table.

**Table 30.3 Clock Supply**

| Explanation | Specification |
|---|---|
| All of each module operations | High speed system clock: CLK_HSB |

### 30.1.4 Interrupt and DMA/DTS Requests

This module has no interrupt and DMA/DTS requests.

### 30.1.5 External Input and Output Pins

This module has no external input/output pins.

# 30.2 Overview

## 30.2.1 Functional Overview

The Data CRC Function B can be used to verify or generate CRC protected data streams of arbitrary length and different bit widths.

- Supported CRC polynomials
  - 32-bit Ethernet CRC
    $04C11DB7_H : X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X^1+1$
  - 16-bit CCITT CRC
    $1021_H : X^{16}+X^{12}+X^5+1$
  - 8-bit SAE J1850 CRC
    $1D_H : X^8+X^4+X^3+X^2+1$
  - 8-bit 0x2F CRC
    $2F_H : X^8+X^5+X^3+X^2+X+1$
- CRC generation to an arbitrary data block length
- After initialization of the DCRB data register, every write access to the DCRB input register generates a new CRC according to the chosen polynomial and the result is stored in the DCRB data register.

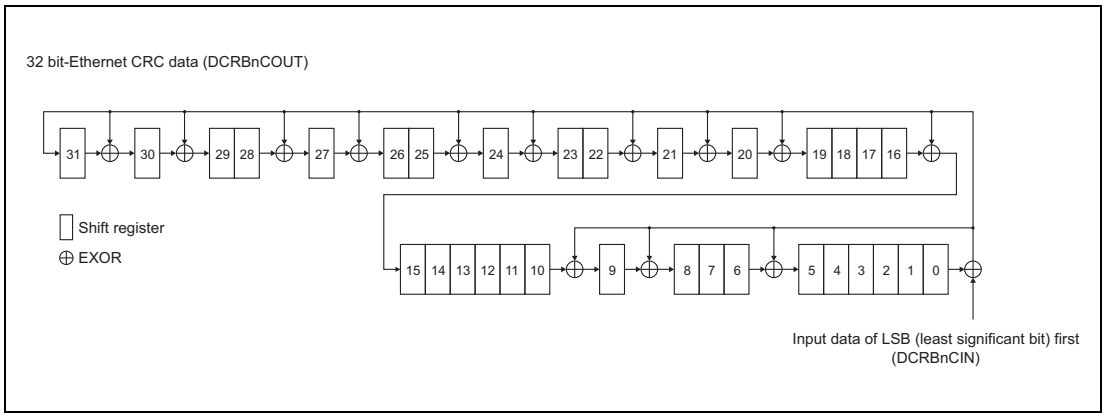



**Figure 30.1 HW implementation of 32-bit Ethernet CRC calculation**

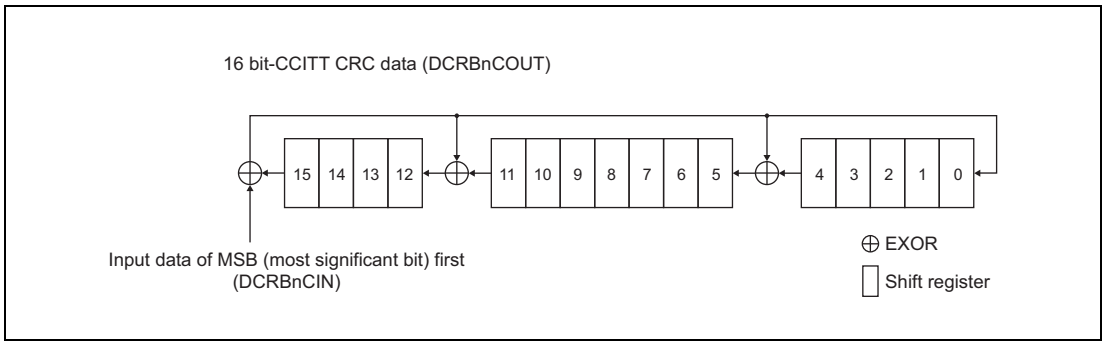



**Figure 30.2 HW implementation of 16-bit CCITT CRC calculation**

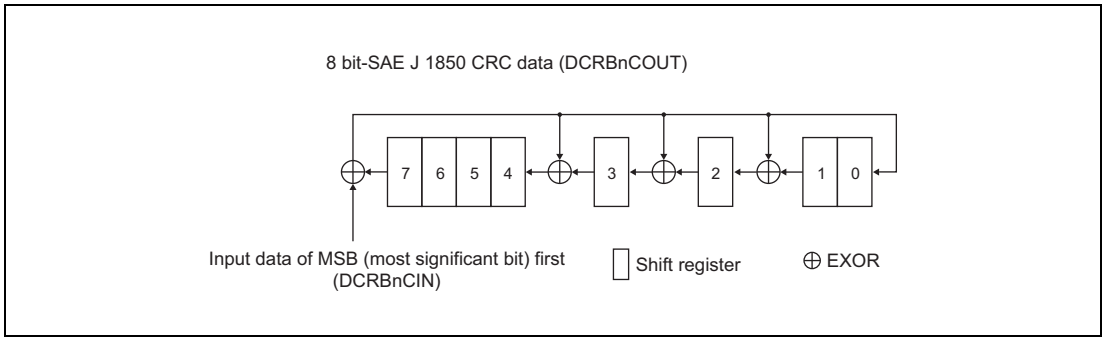



**Figure 30.3 HW implementation of 8-bit SAE J1850 CRC calculation**

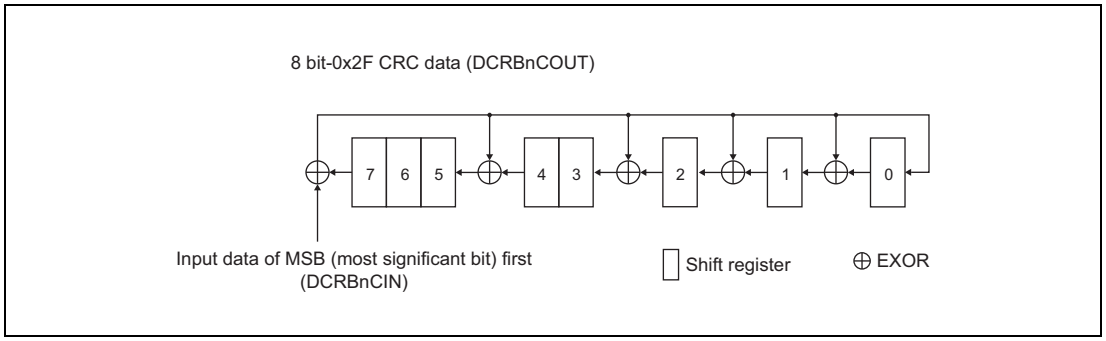



**Figure 30.4 HW implementation of 8-bit 0x2F polynomial CRC calculation**

### 30.2.2 Block Diagram

The following figure shows the block diagram of the Data CRC Function B.

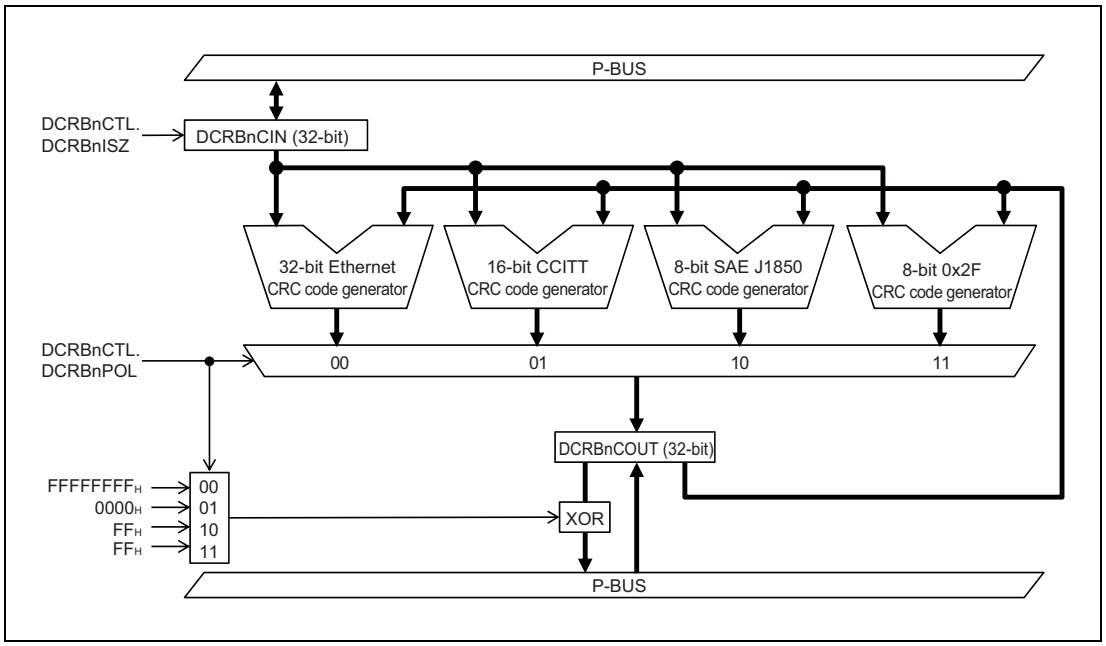



**Figure 30.5 Block diagram of Data CRC Function B**

**Note:** When reading from register DCRBnCOUT, the value is XORed by hardware with the value specified in the AUTOSAR standard for the chosen polynomial.

# 30.3 Registers

## 30.3.1 List of Registers

Table 30.4 List of Registers

| Address Offset*1 | Register Name | Description | Access Width | Value after reset | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | Other |
| $00_H$ | DCRBnCIN | DCRB input register | 32 | 0000 $0000_H$ | — | — |
| $04_H$ | DCRBnCOUT | DCRB data register | 32 | FFFF $FFFF_H$*2 | — | — |
| $20_H$ | DCRBnCTL | DCRB control register | 8 | $00_H$ | — | — |

Note 1. Each base address is described in **Table 30.2**.

Note 2. The read value after reset is 0000 $0000_H$, since the value is XORed by hardware and the 32-bit Ethernet CRC polynomial is selected as the CRC generating function after reset.

Table 30.5 Register Reset Condition

| Register Name | Reset Condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset 1 | System Reset 2 | Application Reset 1 | Limited Reset |
| All registers | √ | √ | √ | √ | — |

### 30.3.2 DCRBnCIN — DCRB input register

This register holds the input data for the CRC calculation. The effective bit width used for CRC calculation must be set by DCRBnCTL.DCRBnISZ[1:0].

When data is written to this register, the CRC code is generated.

The CRC calculation is immediately started after the DCRBnCIN register is written. The DCRBnCOUT register must be initialized, with the initial starting value, before the first data of the data block is written to DCRBnCIN register.

The byte order in DCRBnCIN depends on the selected CRC generating function:

- 32-bit Ethernet CRC polynomial generation (DCRBnCTL.DCRBnPOL[1:0] = 00)
  The byte order is LSB (least significant byte) first, this means that if the CRC input bit width is 8 bits (DCRBnISZ[1:0] = 10B), the input values are in bits 7 to 0 of the DCRBnCIN register and bit 0 (the LSB) is the first bit of the input data.
- 16-bit CCITT CRC polynomial generation (DCRBnCTL.DCRBnPOL[1:0] = 01)
  The byte order is MSB (most significant byte) first, this means that if the CRC input bit width is 8 bits (DCRBnISZ[1:0] = 10B), the input values are in bits 7 to 0 of the DCRBnCIN register and bit 7 (the MSB) is the first bit of the input data.
- 8-bit SAE J1850 CRC polynomial generation (DCRBnCTL.DCRBnPOL[1:0] = 10)
  The byte order is MSB (most significant byte) first, this means that if the CRC input bit width is 8 bits (DCRBnISZ[1:0] = 10B), the input values are in bits 7 to 0 of the DCRBnCIN register and bit 7 (the MSB) is the first bit of the input data.
- 8-bit 0x2F CRC polynomial generation (DCRBnCTL.DCRBnPOL[1:0] = 11)
  The byte order is MSB (most significant byte) first, this means that if the CRC input bit width is 8 bits (DCRBnISZ[1:0] = 10B), the input values are in bits 7 to 0 of the DCRBnCIN register and bit 7 (the MSB) is the first bit of the input data.

**Access:** DCRBnCIN register can be read/written in 32-bit units.

**Address:** <DCRBn_base>

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCRBnCIN | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCRBnCIN | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 30.6 DCRBnCIN register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DCRBnCIN | Input data for CRC calculation.<br>The valid bits are:<br>• For 32 bit effective bit width: DCRBnCIN[31:0]<br>• For 16 bit effective bit width: DCRBnCIN[15:0]<br>• For 8 bit effective bit width: DCRBnCIN[7:0] |

### 30.3.3 DCRBnCOUT — DCRB data register

This register stores the result of the CRC code generated by the 32 bit Ethernet, 16-bit CCITT, 8-bit SAE J1850 or 8-bit 0x2F polynomial.

**Access:** DCRBnCOUT register can be read/written in 32-bit units.

**Address:** <DCRBn_base> +$4_H$

**Value after reset:** FFFF $FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCRBnCOUT | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCRBnCOUT | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 30.7 DCRBnCOUT register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | DCRBnCOUT [31:0] | Result of the CRC code generation.<br>When the 16-bit CCITT polynomial is enabled, the bits 15 to 0 show the CRC result. The bits 31 to 16 are undefined.<br>When 8-bit SAE J1850 or 8-bit 0x2F polynomial is enabled, bits 7 to 0 show the CRC result. The bits 31 to 8 are undefined.<br>On reading these bits, the value read out is the XOR of the value described below and the DCRBnCOUT.DCRBnCOUT bits (this does not affect the DCRBnCOUT.DCRBnCOUT bits).<br>• 32-bit Ethernet polynomial: FFFF $FFFF_H$<br>• 16-bit CCITT CRC polynomial: $0000_H$<br>• 8-bit SAE J1850 polynomial: $FF_H$<br>• 8-bit 0x2F polynomial: $FF_H$<br>Therefore, for the 32-bit Ethernet polynomial, 0000 $0000_H$ is read from this register even in the initial state (FFFF $FFFF_H$). |

**CAUTION**

This register must be initialized with the start value before the first data of the data block is written to DCRBnCIN register.

### 30.3.4 DCRBnCTL — DCRB control register

This register controls the CRC generation process.

**Access:** DCRBnCTL register can be read/written in 8-bit units.

**Address:** <DCRBn_base> +$20_H$

**Value after reset:** $00_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | DCRBnISZ | | — | — | DCRBnPOL | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | R/W | R | R | R/W | R/W |

**Table 30.8 DCRBnCTL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 5, 4 | DCRBnISZ | Specifies the CRC input bit width:<br>00: 32 bit (DCRBnCIN[31:0])<br>01: 16 bit (DCRBnCIN[15:0])<br>10: 8 bit (DCRBnCIN[7:0])<br>11: Setting prohibited |
| 3, 2 | Reserved | When read, the value after reset is read. When writing, write the value after reset. |
| 1, 0 | DCRBnPOL | Specifies the CRC generating function:<br>00: 32-bit Ethernet CRC polynomial generation.<br>The byte order of the DCRBnCIN register is LSB (Least Significant Byte) first, this means that if the CRC input bit width is 8 bits (DCRBnISZ[1:0] = 10B), the input values are in bits 7 to 0 of the DCRBnCIN register and bit 0 (the LSB) is the first bit of the input data.<br>01: 16-bit CCITT CRC polynomial generation.<br>The byte order of the DCRBnCIN register is MSB (Most Significant Byte) first, this means that if the CRC input bit width is 8 bits (DCRBnISZ[1:0] = 10B), the input values are in bits 7 to 0 of the DCRBnCIN register and bit 7 (the MSB) is the first bit of the input data.<br>10: 8-bit SAE J1850 CRC polynomial generation<br>The byte order of the DCRBnCIN register is MSB (Most Significant Byte) first, this means that if the CRC input bit width is 8 bits (DCRBnISZ[1:0] = 10B), the input values are in bits 7 to 0 of the DCRBnCIN register and bit 7 (the MSB) is the first bit of the input data.<br>11: 8-bit 0x2F CRC polynomial generation<br>The byte order of the DCRBnCIN register is MSB (Most Significant Byte) first, this means that if the CRC input bit width is 8 bits (DCRBnISZ[1:0] = 10B), the input values are in bits 7 to 0 of the DCRBnCIN register and bit 7 (the MSB) is the first bit of the input data. |

**NOTE**

After changing the CRC generating function (DCRBnCTL.DCRBnPOL[1:0]) the DCRBnCOUT register must be initialized.

**CAUTIONS**

1. The CRC bit width (DCRBnCTL.DCRBnISZ[1:0]) must be set according to the data block bit width. Switching the CRC bit width is not allowed during processing of a data block (a data block consists of N bytes, half words or words). After the final CRC result is read from DCRBnCOUT register, the bit width can be changed and the DCRBnCOUT register must be initialized with the initial value afterwards.
2. Switching the CRC polynomial (DCRBnCTL.DCRBnPOL[1:0]) is also not allowed during processing of a data block.

## 30.4 Operation

The Data CRC Function B generates a CRC (cyclic redundancy check) of an arbitrary data block length. The data is forwarded to the Data CRC Function in 8-, 16- or 32-bit units. The CRC polynomial can either be selected for 32-bit Ethernet, 16-bit CCITT, 8-bit SAE J1850 or 8-bit 0x2F polynomial CRC, the initial starting value must be set at the DCRBnCOUT register before the first write access to the DCRB input register (DCRBnCIN) is performed.

After the last write access to the DCRBnCIN register is performed, the result can be read-out from the DCRBnCOUT register after one clock period.

The flow chart below shows the CRC generating procedure.

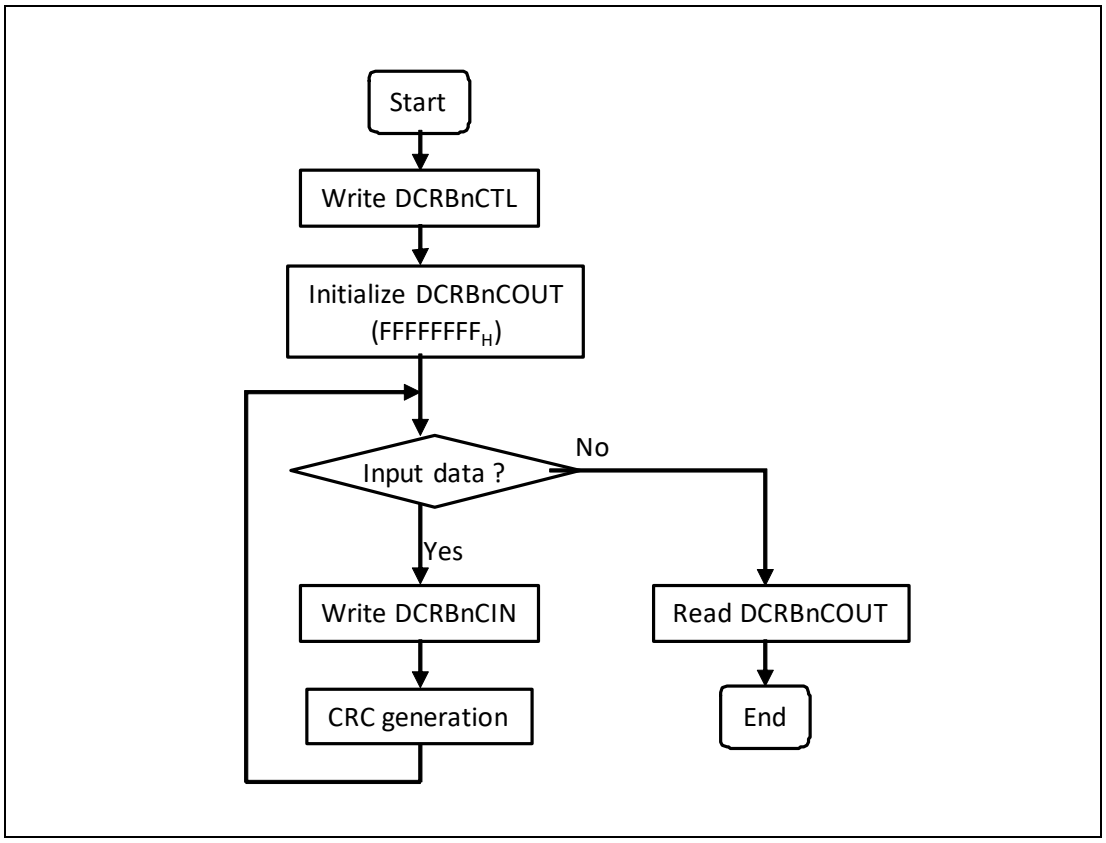



**Figure 30.6 Data CRC Function B flow diagram**

**NOTE**

- The initial value that has to be written into DCRBnCOUT can be chosen by the user. To be compliant to the AUTOSAR standard the initial value has to be chosen according to the value specified for the chosen polynomial.
- When reading from register DCRBnCOUT, the value is XORed by hardware with the value specified in the AUTOSAR standard for the chosen polynomial.
- Generation of CRC code and storage to DCRBnCOUT are done by hardware.
- All registers are accessible by CPU or by CPU-independent read/write access via DMA/DTS. Independent from access of the DCRB via CPU or DMA/DTS the verification of the CRC signature (CRC-check result in DCRBnCOUT) has to be performed by SW.

## 30.5 Difference among P1M-C, P1H-C and P1H-CE

The difference among devices is only the number of units. For details, see **Table 30.1** and **Table 30.2**.

# Section 31 On-Chip Debugging Unit (OCD)

## 31.1 Debug Function

This product has an on-chip debug function. Using an on-chip debug emulator, the micro controller included in the target system debugs programs.

**CAUTION**

**The debug function described in this section is supported by the microcontroller, but whether it can be used or not depends on the debugger. For details of the debugger, see the debugger's user's manual.**

### (1) Debug Interface

This product supports the NEXUS JTAG and Low Pin Debug Interface (4 pins) (hereinafter referred to as LPD (4 pins)) as the debug interfaces.

### (2) Bypass Interface (P1H-CE only)

Emulation device incorporates the AUD-RAM monitor and supports the AUD-RAM monitor interfaces to monitor and tune on-chip RAM data, the peripheral registers, and the like.

### (3) Debug Monitor Function

In debug mode, the monitor program proceeds in the debug-specific area.

Execution of the monitor program enables the following basic debug functions:

- Download of user programs
- Reading and writing of user resources including memory and registers while a user program is suspended.
- Execution of user programs starting from any addresses

### (4) On-chip Break Function

Twelve break points are included in the CPU. Four of them can be designated for any accesses (access address and access data).

### (5) Software Break Function

A software break point can be designated for any addresses on the user programs stored in the RAM.

### (6) Forced Break Function

Execution of a user program can be forcibly suspended.

### (7) Debug Interrupt Interface Function (P1H-CE only)

Execution of a user program can be forcibly suspended by asserting an input to the EVTIZ pin from the outside.

### (8) Forced Reset Function

The micro controller (this product) can be forcibly reset.

#### (9) Real-time RAM Monitor (RRM)

A memory can be read during execution of a program. This read access can minimize the impact on program execution because it uses the debug-specific DMA.

#### (10) Dynamic Memory Modify (DMM)

A memory can be written during execution of a program. This write access can minimize the impact on program execution because it uses the debug-specific DMA.

#### (11) Timer Function

When a 32-bit counter is used, the time period during execution of a user program can be measured on the basis of a debug-specific clock.

#### (12) Mask Function

A reset factor (pin reset, software reset, or ECM reset) can be masked.

#### (13) Event Detection Function

Events can be detected by the following: execution address, access address, access data, range (comparison in size), and sequential execution.

#### (14) Trigger Input Interface (P1H-CE only)

This product incorporates an event trigger input interface to accept external events. It can accept an external event in response to an input from the EVTIZ pin.

#### (15) Trigger Output Interface (P1H-CE only)

An event trigger output interface is included in this product to notify the external debug device of event detection and the like. Output from the EVTOZ pin can output a trigger of event detection to the outside.

#### (16) Hot Plug-in Function

Debugging can be started in normal operating mode without an input of an external reset.

#### (17) Security Function

To prevent the contents of the flash memory from being read by an unauthorized person, a 256-bit ID code (OCD_ID) can be written to the microcontroller. If the code the user inputs when starting a debugger does not match the ID code, the flash memory cannot be accessed.

In addition, the devices supporting hardware security module provides higher security system. Debug functions are under controlled by security module and unauthorized person must communicate with security module for connecting any tools.

See *RH850/P1x-C User's Manual: Hardware (Security)* ***Section 1 Basic Security*** and ***Section 3 ICUMC***.

#### (18) Calibration Function (P1H-CE only)

Emulation of the flash memory and tuning of ROM data can be executed by using the ERAM, the memory for emulation. For details, **Section 31.2, Calibration Function (P1H-CE only)**.

#### (19) Tracing Function

User program execution history, data variation, and the like can be obtained. For details, see **Section 31.3, Trace Control Function**.

## 31.2 Calibration Function (P1H-CE only)

P1H-CE includes the Emulation RAM as an emulation memory for the on-chip flash memory.

### (1) Emulation RAM

This product includes 2-Mbyte emulation RAM.

### (2) Flash Emulation Function

The Emulation RAM can be mapped to any areas in the flash area (up to 32 blocks for each bank).

- Block 0-31 for bank A: 32 Kbytes each
- Block 0-31 for bank B: 32 Kbytes each

### (3) Tuning Function

The ROM data can be dynamically tuned during execution of a user program via the Emulation RAM that been mapped to the flash area.

### (4) Flash Cache Clear Function

The flash cache is cleared when the Emulation RAM mapping is set. This preserves coherency of the contents between the on-chip flash memory or the Emulation RAM and the flash cache memory.

## 31.3 Trace Control Function

This product provides several trace functions including branch PC trace of the CPU, data trace, and DMA data trace.

### (1) Trace Interface: Aurora (P1H-CE only)

To achieve the necessary data throughput and to transfer the trace packages of a Multi-core system without data lost, a differential high-speed serial interface will be supported by the emulation device. This interface is compliant to the IEEE-ISTO 5001-2012 high-speed auxiliary port standard and the Aurora protocol.

### (2) Software Trace

This function enables the obtaining of histories of user program execution, data variation, and the like.

The software trace information can be output via the debug interfaces, LPD (4 pin).

## 31.4 Peripheral Break Control

Peripheral break is a function for stopping the peripheral modules when a user program is halted (at a break point, etc.).

The on-chip modules can be classified into two by its operation at the time of peripheral break as follows:

1. Modules that are unconditionally stopped: WDTA0, WDTA1 and SWDT
2. Modules that can select abeyance or continuation: STM, GTM, CSIH0 to CSIH3 and RLIN30 to RLIN33.

## 31.5 Usage Notes for On-chip Debugging

### (1) Caution of Devices Used for Debugging

Do not mount a device used for debugging on mass-produced products. The number of times data can be written to the flash memory cannot be guaranteed because the memory has already been rewritten.

# Section 32 Flash Memory

## 32.1 Features

- Code flash memory capacity: Up to 10 Mbytes of user area. The devices with multi core (P1H-C and P1H-CE) will follow a banked memory approach.
- Data flash memory capacity: Up to 192 + 32 Kbytes for data area exclusively for ICUMC (P1M-C, P1H-C and P1H-CE)*1.
- Methods of programming :
  - Programming by communicating with the dedicated flash memory programmer via the serial interface (Serial programming).
  - Flash memory programming by a user program (Self-programming).
- Support for security functions to protect against illicit tampering with or reading out of data in flash memory.
- Support for protection functions to protect against erroneous overwriting of the flash memory.
- Support for OTP (one time programming) on Code Flash.
- Support for the detection and correction of errors in the flash memory.
- Support for the BGO (Back Ground Operation) function.
  - Code flash memory can be read while data flash memory is being programmed.
  - While flash memory in one bank is being erased/programmed/read, flash memory in the other bank can be erased/programed/read.
- The Option Bytes register value (some settings of the device) can be configured in the extended area of flash memory.
- Smallest writable unit on Code Flash is 256 bytes, on Data Flash is 4 bytes.
- Smallest erasable unit is a block. On Code Flash 8 Kbytes or 32 Kbytes and on data Flash 64 bytes.

**Note 1.** Can be used as normal data flash in case of non-secure applications.

### 32.1.1 Clock Supply

Clock supply to Flash Control Logic (FACI) is listed in the following table.

**Table 32.1 Clock Supply**

| Unit Name | Clock for the Unit | Supply Clock Name |
|---|---|---|
| FACI | Operation clock | CLK_LSB |

## 32.2 Block Diagram

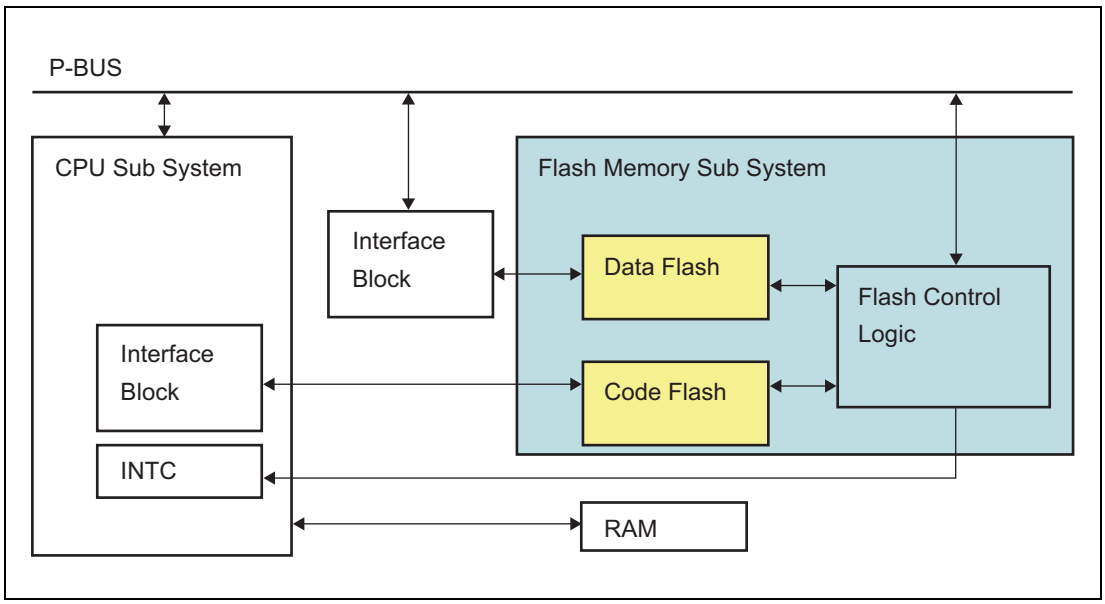



Figure 32.1 Block Diagram

## 32.3 Flash Size

Table 32.2 Flash size

Unit : [Kbytes]

| Name | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|
| Code Flash | 2048 | 4096 | 8192 | 10240 |
| Data Flash*1 | 64 + 32 (ICUMC exclusive)*2 | 128 + 32 (ICUMC exclusive)*2 | 192 + 32 (ICUMC exclusive)*2 | 192 + 32 (ICUMC exclusive)*2 |

Note 1. Data Flash sizes including for all devices the erase counter.

Note 2. Can be used as normal data flash in case of non-secure applications.

## 32.4 Memory Configuration

User area in code flash memory of RH850/P1x-C is divided into 8 Kbytes or 32 Kbytes blocks, which can be erased individually.

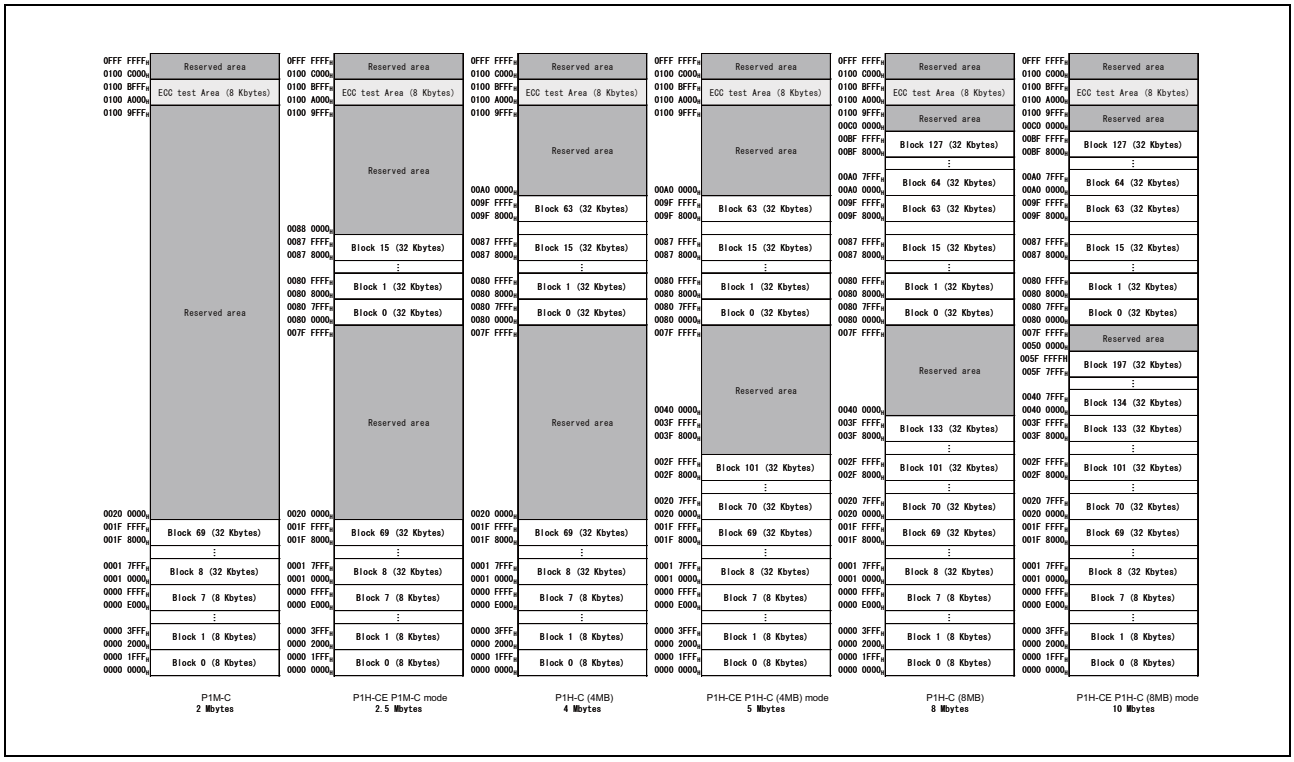



**Figure 32.2 Code Flash Memory Mapping**

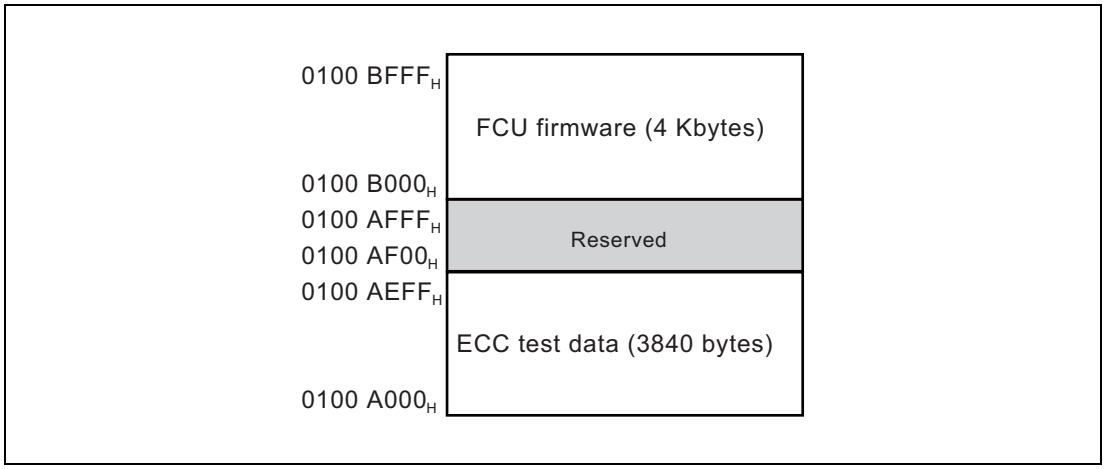
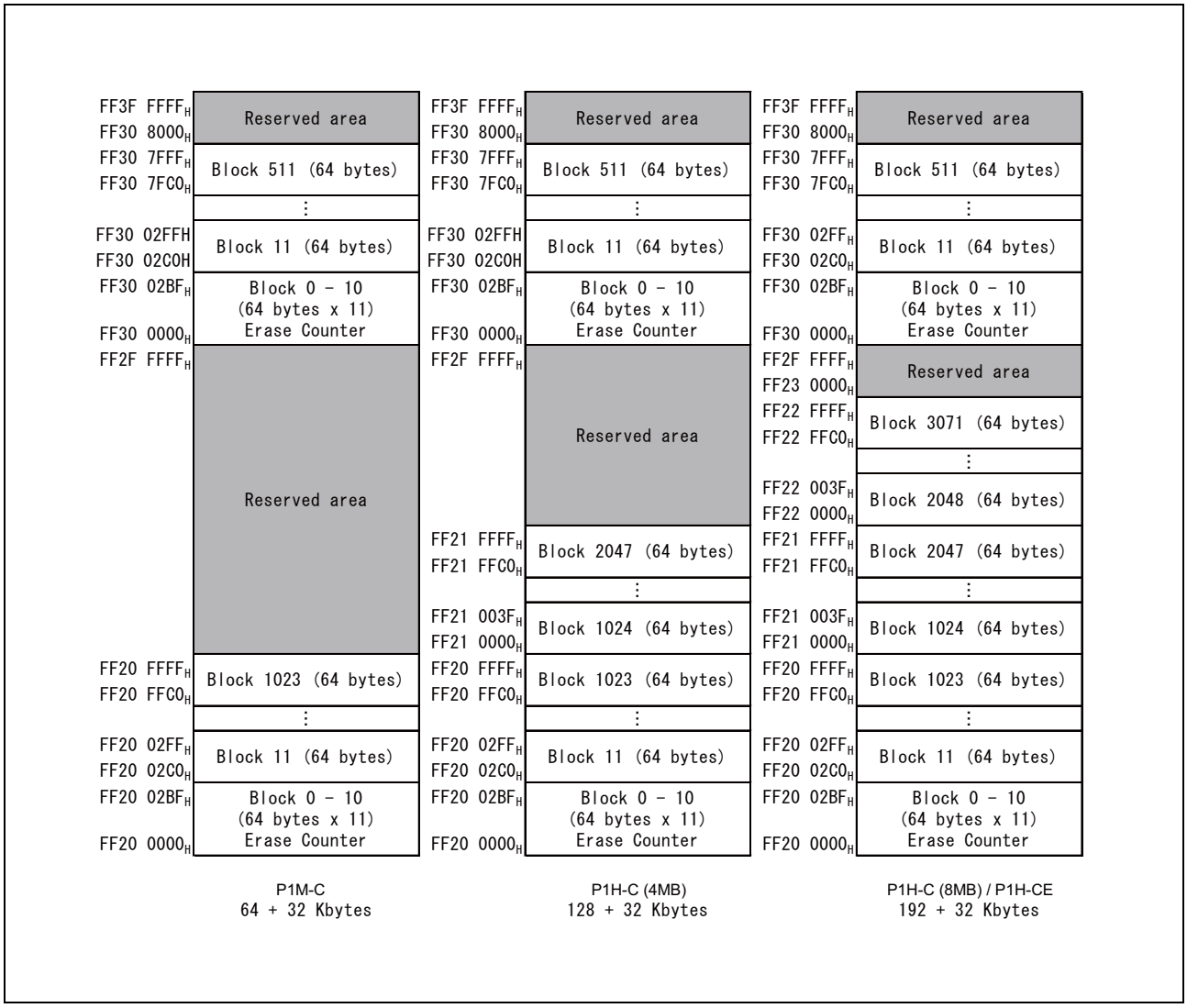



**Figure 32.3 Detail of ECC test Area**

Data area in data flash memory of RH850/P1x-C is divided into 64 bytes blocks, which can be erased individually.



# 32.5 Registers

## 32.5.1 Register Base Address

Table 32.3 Register Base Address

| Instance Name | Base Address |
|---|---|
| DCIB0 | DCIB0_base = FFC5 9800$_H$ |
| DCIB1 | DCIB1_base = FFC5 9900$_H$ |
| SCDS | SCDS_base = FFCD 0000$_H$ |
| SYSCTL | SYSCTL_base = FFF8 0000$_H$ |

## 32.5.2 List of Registers

Table 32.4 List of Registers

| Register Name | Abbreviation | R/W | Value after reset | Address | Access Size | Access Protection |
|---|---|---|---|---|---|---|
| Data Flash Memory Read Cycle Setting Register0 | EEPRDCYCL0 | R/W | 0F$_H$ | DCIB0_base + 0010$_H$ | 8 | PBG5#0. PG5-DCIB0[*1] |
| Data Flash Memory Read Cycle Setting Register1 | EEPRDCYCL1 | R/W | 0F$_H$ | DCIB1_base + 0010$_H$ | 8 | PBG5#0. PG5-DCIB1[*1] |
| Reset Vector for PE1/PE2 register | GREG8 | R | XXXX XXXX$_H$ | SCDS_base + 0020$_H$ | 32 | PBG4#1. PG4-FLASH |
| Option Byte 0 | OPBT0 | R | XXXX XXXX$_H$ | SCDS_base + 0030$_H$ | 32 | PBG4#1. PG4-FLASH |
| Option Byte 1 | OPBT1 | R | XXXX XXXX$_H$ | SCDS_base + 0034$_H$ | 32 | PBG4#1. PG4-FLASH |
| Option Byte 2 | OPBT2 | R | XXXX XXXX$_H$ | SCDS_base + 0038$_H$ | 32 | PBG4#1. PG4-FLASH |
| Option Byte 13 | OPBT13 | R | XXXX XXXX$_H$ | SCDS_base + 0064$_H$ | 32 | PBG4#1. PG4-FLASH |
| Option Byte 14 | OPBT14 | R | XXXX XXXX$_H$ | SCDS_base + 0068$_H$ | 32 | PBG4#1. PG4-FLASH |
| Option Byte 15 | OPBT15 | R | XXXX XXXX$_H$ | SCDS_base + 006C$_H$ | 32 | PBG4#1. PG4-FLASH |
| Product Name Storage Register 1 | PRDNAME1 | R | XXXX XXXX$_H$ | SCDS_base + 00D0$_H$ | 32 | PBG4#1. PG4-FLASH |
| Product Name Storage Register 2 | PRDNAME2 | R | XXXX XXXX$_H$ | SCDS_base + 00D4$_H$ | 32 | PBG4#1. PG4-FLASH |
| Product Name Storage Register 3 | PRDNAME3 | R | XXXX XXXX$_H$ | SCDS_base + 00D8$_H$ | 32 | PBG4#1. PG4-FLASH |
| Product Name Storage Register 4 | PRDNAME4 | R | XXXX XXXX$_H$ | SCDS_base + 00DC$_H$ | 32 | PBG4#1. PG4-FLASH |
| FHVE15 control register | FHVE15 | R/W | 0000 0000$_H$ | SYSCTL_base + A430$_H$ | 32 | PBG4#0. PG4-SC3 |
| FHVE3 control register | FHVE3 | R/W | 0000 0000$_H$ | SYSCTL_base + 2410$_H$ | 32 | PBG4#0. PG4-SC3 |

Note 1. See *RH850/P1x-C User's Manual: Hardware (Security)* ***Section 3 ICUMC***.

### 32.5.3 Register Reset Condition

Table 32.5 Register Reset Condition

| Register Name | Reset Condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset1 | System Reset2 | Application Reset | Limited Reset |
| EEPRDCYCL0 | √ | √ | √ | √ | |
| EEPRDCYCL1 | √ | √ | √ | √ | |
| FHVE15 | √ | √ | √ | √ | |
| Other Registers | √ | √ | √ | | |

## 32.6 Operating Modes Associated with Flash Memory

**Figure 32.5** is a diagram of the mode transitions associated with the flash memory. For the procedures for setting the modes, see **Section 5, Operating Modes**

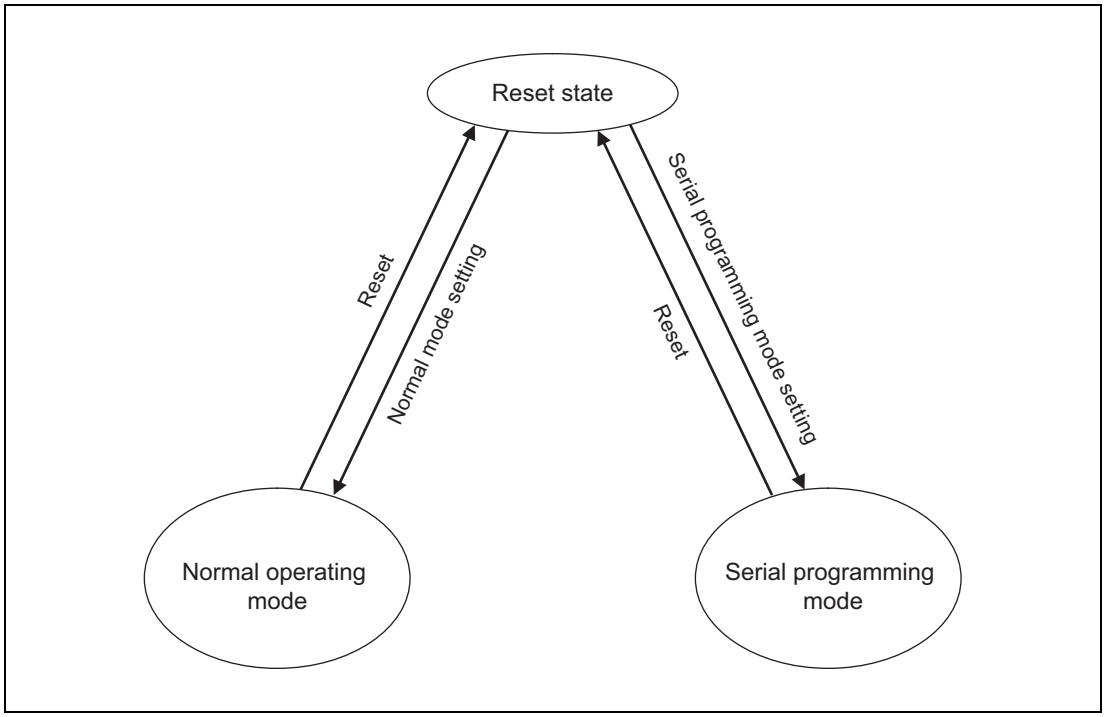



**Figure 32.5 Mode Transition Associated with Flash Memory**

The flash memory areas which are programmable and erasable and the boot program after a reset depend on the selected mode. The differences between modes are indicated in **Table 32.6**.

**Table 32.6 Differences between Modes**

| Item | Normal Operating Mode | Serial Programming Mode |
|---|---|---|
| Programmable and erasable area | User area<br>Data area | User area<br>Data area |
| Boot program at a reset | Program in user area | Firmware program for serial programming |

## 32.7 Functional Overview

On-chip flash memory of RH850/P1x-C can be programmed regardless of before and after the mounting to the target system with the programming function that employs the dedicated flash memory programmer (serial programming).

In addition, a security function that prohibits the programming of a user program written in on-chip flash memory is supported to address the protection against falsification of programs by outsiders.

The programming function using a user program (self-programming) is the method suitable for applications where modifying the program after the production or shipment of the target system is expected. A protection function for secure programming to flash memory area is also supported. Furthermore, programming can be conducted under various conditions, such as in parallel with communicating with outside, by utilizing the support for interrupt processing during self-programming. **Table 32.7** gives an overview of the methods of programming and the corresponding operating modes.

**Table 32.7 Programming Methods**

| Programming Method | Functional Overview | Operating mode |
|---|---|---|
| Serial programming | A dedicated flash-memory programmer is capable of on-board programming the flash memory after the device is mounted on the target system. | Serial programming mode |
| | A dedicated flash memory programmer and dedicated programming adapter board allow off-board programming of the flash memory, i.e. programming of the device before it is mounted on the target system. | |
| Self-programming | Flash memory can be programmed by executing a user program preprogrammed in code flash memory with serial programming.<br>When data flash memory is being programmed with self-programming, the BGO function enables instruction fetch and data read from code flash memory. Thus, data flash memory can be programmed by executing a program on code flash memory prepared for flash programming.<br>When code flash memory is being programmed with self-programming,instruction fetch and data read from code flash memory for a bank during the programing are prohibited. This programming needs to be carried out by executing a program prepared for flash programming that has been transferred to Local RAM in advance or a program on a different bank. | Normal operating mode |

**Table 32.8** lists the functions of the on-chip flash memory. Serial programmer commands realize serial programming, while reading of the on-chip flash memory by a library function or the user program realizes self-programming.

**Table 32.8 Basic Functions at a Glance**

| Function | Description in Overview | Level of Support (√: Supported, △: Conditionally Supported, ×: Not Supported) | |
|---|---|---|---|
| | | Serial programming | Self-programming |
| Blank checking*1 | This is used to check a specified block to ensure that writing to it has not already proceeded. Results of reading from code flash memory and data flash memory to which nothing has been written after erasure are not guaranteed, so use blank checking to confirm that writing to memory has not proceeded after erasure. | √ | √ |
| Block erasure | This is for erasing the contents of a specified block of memory. | √ | √ |
| Programming*1 | This is for writing to a specified address. | √ | √ |
| Verification and checksum | Data that are read out from flash memory are compared with data transferred from the flash memory programmer. | √ | × (reading of data by the user program is possible) |
| Reading | Data that have been written to the flash memory are read out. | √ | √ |
| Setting for OTP (one-time programming) | A specified block of code flash memory is set for OTP (OTP can only be set, that is, it is not possible to release a block's OTP setting). | √ | √ |
| Setting an ID | An ID setting is made for use in programming of the code flash memory by self-programming. | × | √ |
| Security settings | Security settings are for use in serial programming. | √ | △ (only when setting is prohibited after being permitted) |
| Protection settings | Lock bits for all blocks of code flash memory and overall of data flash memory are provided. | √ | √ |
| Setting of option bytes | Option bytes are set to change them from the value after reset for the RH850/P1x-C. | √ | √ |
| Clearing the configuration | ID setting, security settings, protection settings, and option byte settings are cleared. | √ | × |

Note 1. Data Flash Bank A consists of each 64K-byte area. "Program", "Blank Check" command processing in the case of Data Flash cannot cross over address boundary between different 64K Byte area.

For details on serial programming, see "*PG-FP5 Flash Memory Programmer User's Manual*" and "*Renesas Flash Programmer Flash Programming Software User's Manual*".

For details on self-programming, see the user's manuals for the code flash library and data flash library which this device targets.

The OTP setting is security functions for use with serial programming and self-programming and authentication of the ID code is security functions for use with self-programming.

In serial programming, prohibiting connection of a serial programmer and prohibition of commands (for block erasure, programming, and reading) are available for use as security functions.

**Table 32.9 Summary of Security Functions**

| Function | Description |
|---|---|
| OTP | OTP can be individually set for each block of the user area of code flash memory. When the OTP setting is made for an area, programming by serial programming and by self programming is prohibited. Once set, the OTP setting cannot be released. Furthermore, since execution of the configuration clearing command is prohibited for any area for which OTP has been set, changing a security setting from “prohibited” to “permitted” is not possible. |
| ID authentication | The result of ID authentication can be used to control enabling of self-programming. The code flash memory cannot be programmed by self-programming without ID authentication. |
| Prohibition of connection of a serial programmer | The connection of a serial programmer for serial programming is prohibited. Since execution of the configuration clearing command is also prohibited when the connection of a serial programmer is prohibited, changing a security setting from “prohibited” to “permitted” is not possible. |
| Prohibition of block erasure commands | Block erasure commands at the time of serial programming are prohibited. Since execution of the configuration clearing command is also prohibited when block erasure commands are prohibited, changing a security setting from “prohibited” to “permitted” is not possible. |
| Prohibition of programming commands | Block erasure commands and programming commands at the time of serial programming are prohibited. Only through execution of the configuration clearing command can the prohibition be lifted. |
| Prohibition of read commands | Read commands at the time of serial programming are prohibited. Only through execution of the configuration clearing command can the prohibition be lifted. |

Table 32.10 Available Operations and Security Settings (1/2)

| Function | All Security Settings and Erasure, Programming, and Read Operations (√: Executable, ×: Not Executable, —: Not Supported) | | Point for Caution Regarding the Security Setting | |
|---|---|---|---|---|
| | Serial programming | Self programming | Serial programming | Self programming |
| OTP | • Areas for which OTP is set<br>– Block erasure commands: ×<br>– Programming commands: ×<br>– Read commands: √<br>• Areas for which OTP is not set<br>– Block erasure commands: √<br>– Programming commands: √<br>– Read commands: √ | • Areas for which OTP is set<br>– Block erasure: ×<br>– Programming: ×<br>– Reading: √<br>• Areas for which OTP is not set<br>– Block erasure: √<br>– Programming: √<br>– Reading: √ | • The OTP setting cannot be released.<br>• Execution of the configuration clearing command is not possible. | The OTP setting cannot be released. |
| ID authentication | ID authentication is not supported | • When the ID codes do not match<br>[Code flash memory]<br>– Block erasure: ×<br>– Programming: ×<br>– Reading: √<br>[Data flash memory]<br>– Block erasure: √<br>– Programming: √<br>– Reading: √<br>• When the ID codes match<br>– Block erasure: √<br>– Programming: √<br>– Reading: √ | | ID authentication is always in effect. |
| Prohibition of the connection of a serial programmer | • Block erasure commands: ×<br>• Programming commands: ×<br>• Read commands: × | • Block erasure: √<br>• Programming: √<br>• Reading: √ | Since execution of the configuration clearing command is prohibited, initialization of the setting for prohibition is not possible. | Initialization of the settings prohibited is impossible because the configuration clearing command is not supported. |
| Prohibition of block erasure commands | • Block erasure commands: ×<br>• Programming commands: √<br>• Read commands: √ | • Block erasure: √<br>• Programming: √<br>• Reading: √ | • Since execution of the configuration clearing command is prohibited, initialization of the setting for prohibition is not possible.<br>• The setting for prohibition of serial programmer connection is not available. | Initialization of the settings prohibited is impossible because the configuration clearing command is not supported. |
| Prohibition of programming commands | • Block erasure commands: ×<br>• Programming commands: ×<br>• Read commands: √ | • Block erasure: √<br>• Programming: √<br>• Reading: √ | • Executing the configuration clearing command only can initialize the settings prohibited. | Initialization of the settings prohibited is impossible because the configuration clearing command is not supported. |
| Prohibition of read commands | • Block erasure commands: √<br>• Programming commands: √<br>• Read commands: × | • Block erasure: √<br>• Programming: √<br>• Reading: √ | | |

**Table 32.10 Available Operations and Security Settings (2/2)**

| Function | All Security Settings and Erasure, Programming, and Read Operations (√: Executable, ×: Not Executable, —: Not Supported) | | Point for Caution Regarding the Security Setting | |
|---|---|---|---|---|
| | Serial programming | Self programming | Serial programming | Self programming |
| Lock bit | • Areas for which lock bit is set<br>– Block erasure commands: ×<br>– Programming commands: ×<br>– Read commands: √<br>• Areas for which lock bit is canceled or not set<br>– Block erasure commands: √<br>– Programming commands: √<br>– Read commands: √ | • Areas for which lock bit is set<br>– Block erasure: ×<br>– Programming: ×<br>– Reading: √<br>• Areas for which lock bit is canceled or not set<br>– Block erasure: √<br>– Programming: √<br>– Reading: √ | The cancelation of lock bit is protected by basic hardware protection.<br>regarding of basic hardware protection, see relating chapter. | The cancelation of lock bit is protected by basic hardware protection.<br>regarding of basic hardware protection, see relating chapter. |

**Table 32.11 Summary of Protection Functions**

| Function | Description |
|---|---|
| Hardware protection | The level on the FLMD pin can be set to prohibit programming and erasure of the code flash memory.<br>FLMD0 = 0: Programming prohibited<br>FLMD0 = 1: Programming permitted |
| Variable reset vector | The protection settings include control of the reset vector. As shown in **Figure 32.6**, after programming of a new boot program while leaving the existing boot program in place, changing the reset vector is a safe way to change to the area holding the new boot program. |

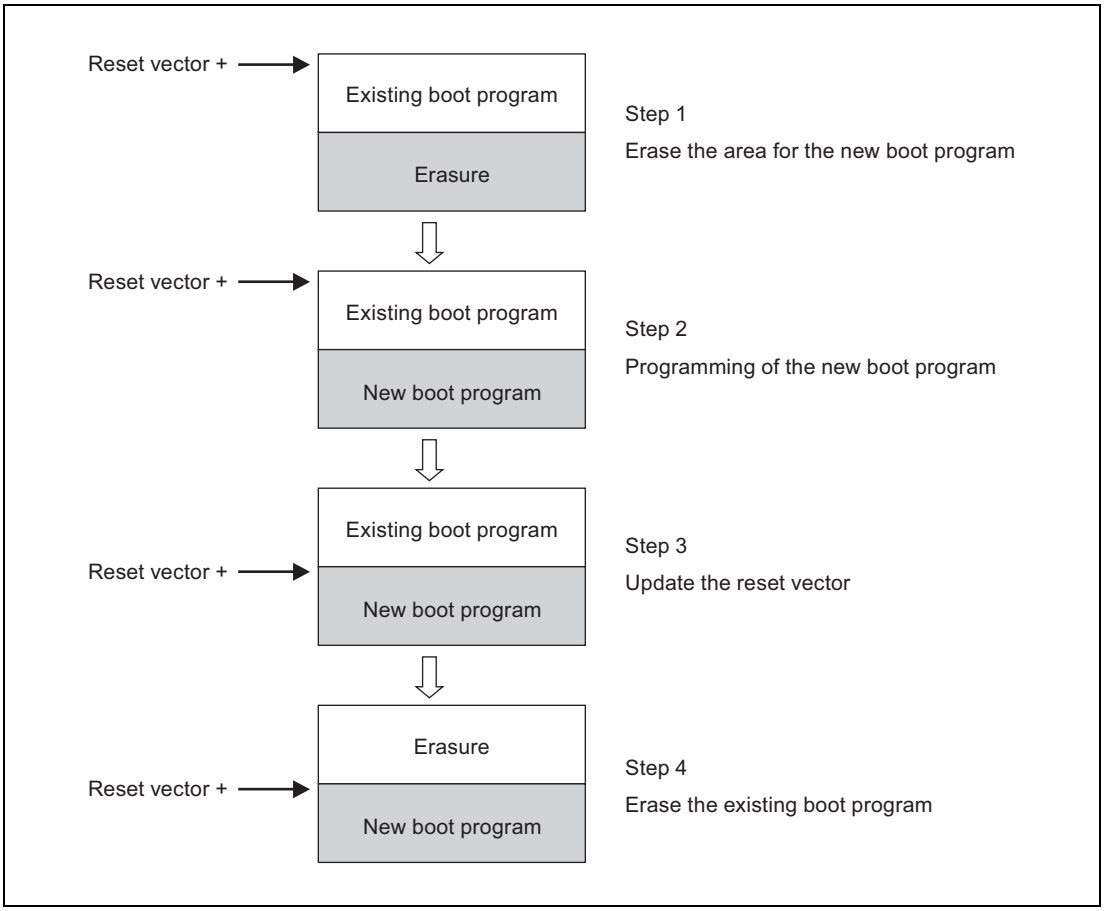



**Figure 32.6 Utilizing the variable reset vector function to update the boot program**

## 32.8 Communications Mode

### 32.8.1 One-wire UART as an Asynchronous Flash Programming Interface

To use a 1-wire UART as a single-wire asynchronous serial programming interface, connect the following pins and the flash memory programmer.

- FLSCI3RXD, FLSCI3TXD (FPDR)/JP0_0: Receive data input/transmit data output

### 32.8.2 Two-wire UART as an Asynchronous Flash Programming Interface

To use a 2-wire UART as a double-wire asynchronous serial programming interface, connect the following pins and the flash memory programmer.

- FLSCI3RXD (FPDR)/JP0_0: Receive data input
- FLSCI3TXD (FPDT)/JP0_1: Transmit data output

### 32.8.3 CSI as a Synchronous Flash Programming Interface

To use a CSI as a synchronous serial programming interface, connect the following pins and the flash memory programmer.

- FLSCI3RXD (FPDR)/JP0_0: Receive data input
- FLSCI3TXD (FPDT)/JP0_1: Transmit data output
- FLSCI3SCKI (FPCK)/JP0_2: Serial clock input

The flash memory programmer outputs the serial data clock signal (SCK) and the microcontroller operates as a slave.

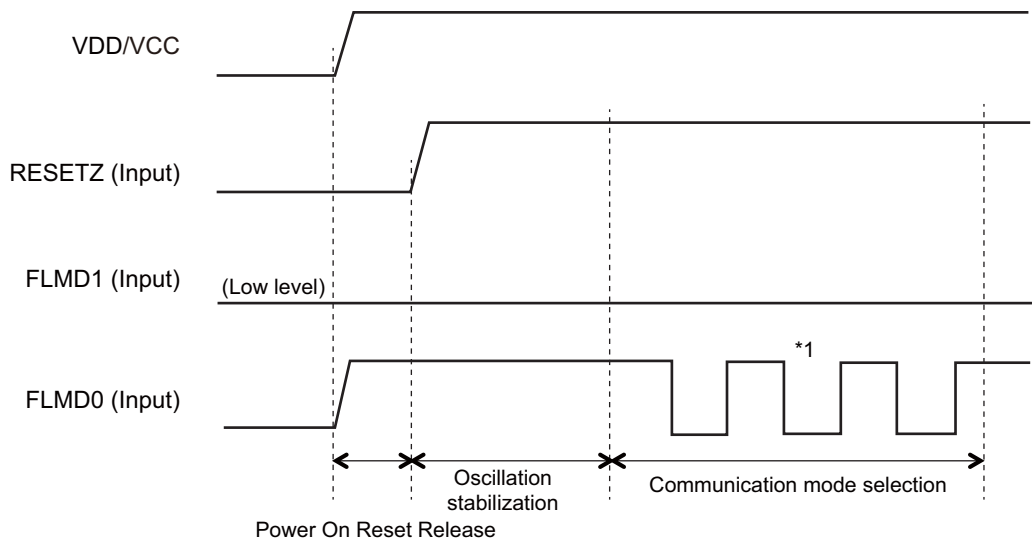
**NOTE**

For details on the flash programming software (Renesas Flash Programmer), see the *Renesas Flash Programmer Flash Programming Software User's Manual*.

### 32.8.4 Selecting the Communications System

In the RH850/P1x-C, the communications system is selected by the input of pulses (up to 7) to the FLMD0 pin after the chip shifts to the flash memory programming mode. These pulses are generated by the dedicated flash memory programmer.

The relationship between the communications system and the number of pulses is shown below.




Note 1. The number of input clocks depends on the communications system as shown below.

| Pulses on FLMD0 | Communications System | Notes |
|---|---|---|
| 0 | 2 wire UART | |
| 7 | 1 wire UART | |
| 3 | CSI | |
| | Others | Setting prohibited |

**Figure 32.7 Selecting the Communications Mode**

## 32.9 Programming with Serial Programmer

Flash memory can be programmed in serial programming mode with the dedicated flash memory programmer.

#### Serial Programming

In serial programming, the microcontroller is mounted on the board. Mounting the connecter on the board allows the flash memory programmer to program the target microcontroller.

### 32.9.1 Programming Environment

The figure below shows the environment recommended for programming data to flash memory in the microcontroller.

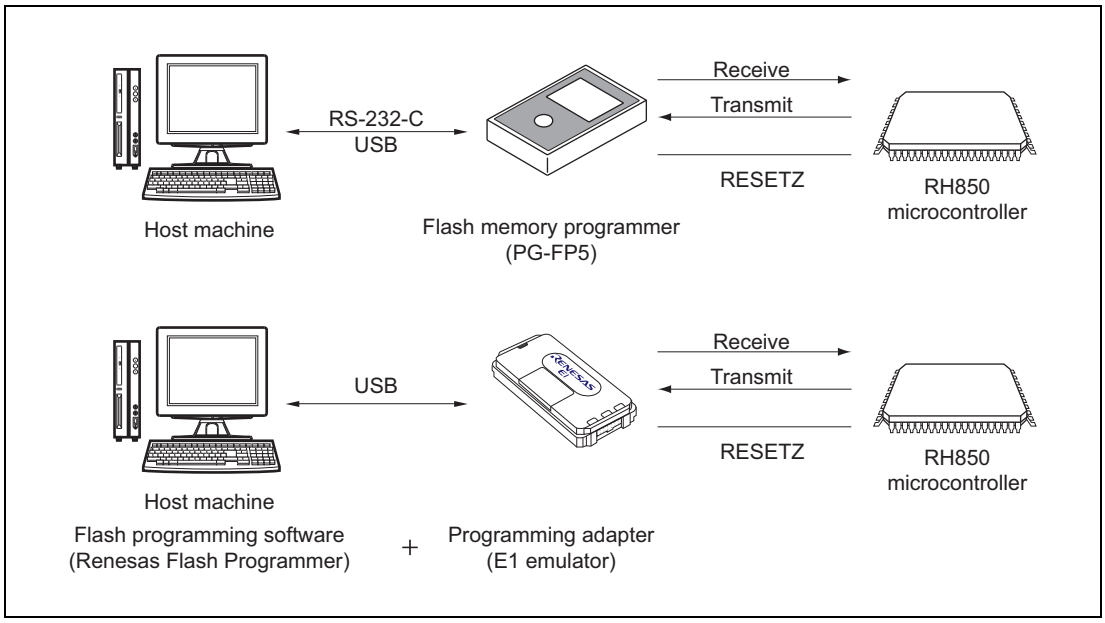



**Figure 32.8 Environment for Programming Flash Memory**

Using PG-FP5 flash memory programmer or Renesas Flash Programmer flash programming software (running on the host machine) in combination with E1 emulator used as the programming adaptor, you can easily conduct programming operations, such as erasure, programming, and verification, to Renesas Electronics' microcontroller with on-chip flash memory mounted on the user's board without the need for unmounting it.

PG-FP5 flash memory programmer supports programming from the host machine or programming in the standalone mode. The flash programming software (Renesas Flash Programmer) supports programming from the host machine.

**NOTE**

See *PG-FP5 flash memory programmer user's manual* for the details of PG-FP5, and *Renesas Flash Programmer flash programming software user's manual* for the details of Renesas Flash Programmer flash programming software.

# 32.10 Programming with Self-programming

## 32.10.1 Overview

RH850/P1x-C supports the flash memory programming of user program itself.

When programming data flash memory, you can execute a program on code flash memory prepared for flash programming using the BGO function to program data flash memory. Instead, you can also execute a program prepared for flash programming that has been transferred to Local RAM in advance to program data flash memory.

In addition, you can execute a program prepared for flash programming that has been transferred to Local RAM in advance to program code flash memory.

You can also execute a program from the different bank to program code flash memory.

## 32.10.2 BGO Function

When the combination of the flash memory to be erased/programmed and the flash memory to be read meets the conditions in the following list, the BGO function can be used.

**Table 32.12 Conditions Required to Use BGO Function for read**

| Area to Be Erased/Programmed | Area to Be Read |
|---|---|
| Data flash memory | Code flash memory |
| Flash memory for a bank | Flash memory for the another bank |

In addition, it is possible to operate programming or erasure in other bank during the programming or erasure in a bank for between different banks.

**Table 32.13 Conditions Required to Use BGO Function for programming/erasure**

| Area to Be Programmed/erasure | Area to Be Programmed/erasure |
|---|---|
| Flash memory for a bank | Flash memory for the another bank |

### 32.10.3 Enabling of Self-Programming

The self-programming function can be activated in normal operating mode.

Erasure and programming of the flash memory by the self-programming function is enabled by making the FLMD0 pin high level.

This prevents unnecessary overwriting of the program if the device operates incorrectly.

The FLMD0 pin is made high level by using one of the following methods.

- The FLMD0 pin is externally pulled up.
- The FLMD0 pin is pulled up by the FLMD control register.

The outline of the FLMD control register is described in **Section 32.10.3.1, FLMDCNT — FLMD control register**

**Table 32.14 List of Enabling of Self-Programming Register**

| Module Name | Register Name | Abbreviation | R/W | Value after Reset | Address | Access Size | Access Protection |
|---|---|---|---|---|---|---|---|
| FLMD | FLMD control register | FLMDCNT | R/W | 0000 $0000_H$ | FFA0 $0000_H$ | 32 | PBG4#1. PG4-FLMD1 (PBG4#2. PG4-FLMD2*[1]) |
| FLMD | FLMD Protection Command Register | FLMDPCMD | W | XXXX $XXXX_H$ | FFA0 $0004_H$ | 32 | PBG4#1. PG4-FLMD1 (PBG4#2. PG4-FLMD2*[1]) |
| FLMD | FLMD Protection Error Status Register | FLMDPS | R | 0000 $0000_H$ | FFA0 $0008_H$ | 32 | PBG4#1. PG4-FLMD1 (PBG4#2. PG4-FLMD2*[1]) |

Note 1. See *RH850/P1x-C User's Manual: Hardware (Security)* ***Section 3 ICUMC***.

### 32.10.3.1 FLMDCNT — FLMD control register

This register specifies the internal pull-up or pull-down of the FLMD0 pin.

**Access:** This register can be read/written in 32-bit units.
Writing to this register is protected by a special sequence of instructions by using the protection command register FLMDPCMD.

**Address:** FFA0 $0000_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FLMDPUP |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 32.15 FLMDCNT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | FLMDPUP | FLMD0 Pin Software Control<br>0: Pull-down selected<br>1: Pull-up selected |

### 32.10.3.2 FLMDPCMD — FLMD Protection Command Register

FLMDPCMD is a protection command register for the FLMDCNT register.

**Access:** This register is a write-only register that can be written in 32-bit units.

**Address:** FFA0 0004$_H$

**Value after reset:** Undefined

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | FLMDPC[7:0] | | | | | | | |
| Value after reset | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

**Table 32.16 FLMDPCMD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 8 | Reserved | When writing, write the prescribed value. |
| 7 to 0 | FLMDPC[7:0] | Protection command register bits that enable writing to self-programming protection cluster registers |

### 32.10.3.3 FLMDPS — FLMD Protection Error Status Register

This register is used to verify whether the write-protected register (FLMDCNT) has been successfully written or not.

**Access:** This register is a read-only register that can be read in 32-bit units.

**Address:** FFA0 0008$_H$

**Value after reset:** 0000 0000$_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FLMDPRERR |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 32.17 FLMDPS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | When writing, write the value after reset. |
| 0 | FLMDPRERR | Indicates whether the write-protected register (FLMDCNT) has been successfully written.<br>0: Write operation successful<br>1: Write operation failed |

### 32.10.4 Procedure for Writing to a Write-Protected Register

FLMDCNT register is Write-Protected Register.

The write protection procedure is based on the following state machine.

Step 1:

When write access to the protection command register is completed, the error flag is cleared to move on to step 2.

Unless the write value is 0000 $00A5_H$, an error flag is set in the protection status register and the state of step 1 is held.

Step 2:

Write access to the protected register should be completed (write data should be the expected value) to move on to step 3. If an unprotected register is accessed, an error flag is set to return to step 1.

Step 3:

Write access to the protected register should be completed (write data should be the reversed value of the expected one) to move on to step 4.

If the write data is not the reversed value or if an unprotected register is accessed, an error flag is set to return to step 1.

Step 4:

When write access to the protected register is completed (write data is the expected value), the write signal of the protected register is activated and writing is completed.

If write data is not the expected one or if an unprotected register is written, an error flag is set to return to step 1.

# 32.11 Flash Memory Read

## 32.11.1 Code Flash Memory Read

Special settings are not required to read code flash memory in normal mode. Data can simply be read out through access to addresses in the code flash memory.

Reading from an area of code flash memory that has been erased but not yet been programmed again (i.e. that is in the non-programmed state) can lead to the detection of an ECC error and generation of the corresponding exception. Furthermore, since the values of data cannot be guaranteed when an ECC error has been generated, use blank checking when you need to confirm that an area is in the non-programmed state.

## 32.11.2 Data Flash Memory Read

Configure the number of read cycles in the EEPRDCYCL0, 1 registers prior to reading data from data flash memory in normal mode.
Once the setting for the number of cycles is made, data can simply be read out through access to addresses in the data flash memory.
Values read from data flash memory that has been erased but not yet been programming again are undefined. Use blank checking when you need to confirm that an area is in the non-programmed state.

EEPRDCYCL0 set the read cycles for data flash Bank A and EEPRDCYCL1 is for Bank B

### 32.11.2.1 EEPRDCYCL0, 1 — Data Flash Memory Read Cycle Setting Register

This register sets the read cycle of data flash memory.

**Access:** This register can be read/written in 8-bit units.

**Address:** EEPRDCYCL0: FFC5 $9810_H$
EEPRDCYCL1: FFC5 $9910_H$

**Value after reset:** $0F_H$

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | FRDCYCLD[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R/W | R/W | R/W | R/W |

**Table 32.18 EEPRDCYCL register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 to 4 | Reserved | (The write value must be 1.) |
| 3 to 0 | FRDCYCLD[3:0] | Number of Data Flash Memory Read Cycles<br>Data flash memory is read in setting value + 1 cycles.<br>0000: Read cycle 1<br>0001: Read cycle 2<br>0010: Read cycle 3<br>0011: Read cycle 4<br>0100: Read cycle 5<br>0101: Read cycle 6<br>0110: Read cycle 7<br>0111: Read cycle 8<br>1000: Read cycle 9<br>Other the above: Read cycle 10<br>The read cycle must be configured to satisfy the below condition<br>Read cycle ≥ (CLK_LSB[MHz]/10) |

## 32.11.3 Registers Related to Write and Erase Protect of Flash Memory

### 32.11.3.1 FHVE15 — FHVE15 Control Register

FHVE15 is a readable/writable register for software protection of flash memory against programming, erasure and blank checking. Set both the FHVE15 and FHVE3 registers in the programmable, erasable and blank-checkable state ($0000\ 0001_H$) to program, erase or blank check flash memory.

**Access:** This register can be read/written in 32-bit units.

**Address:** $FFF8\ A430_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FHVE15CNT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 32.19 FHVE15 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | (The write value must be 1.) |
| 0 | FHVE15CNT | 0: Programming, erasure and blank checking are disabled.<br>1: Programming, erasure and blank checking are enabled. |

### 32.11.3.2 FHVE3 — FHVE3 Control Register

FHVE3 is a readable/writable register for software protection of flash memory against programming, erasure and blank checking. Set both the FHVE15 and FHVE3 registers in the programmable, erasable and blank-checkable state ($0000\ 0001_H$) to program, erase or blank check flash memory.

**Access:** This register can be read/written in 32-bit units.

**Address:** $FFF8\ 2410_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FHVE3 CNT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

**Table 32.20 FHVE3 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 1 | Reserved | (The write value must be 1.) |
| 0 | FHVE3CNT | 0: Programming, erasure and blank checking are disabled.<br>1: Programming, erasure and blank checking are enabled. |

## 32.11.4 Register Related to Reset Vector

### 32.11.4.1 GREG8 — Reset Vector for PE1/PE2 register

**Access:** This register can be read in 32-bit units.

**Address:** FFCD $0020_H$

**Value after reset:** Specified by the user

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Reset Vector 0*1 | | | | | | | | | | | | | | | |
| Value after reset | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Reset Vector 0*1 | | | | | | | | | | | | | | | |
| Value after reset | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. These values are depended on RESV bits of option bytes in the flash memory that are specified by the user.

**Table 32.21 GREG8 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | Reset Vector 0 | Reset vector for PE1/PE2 |

## 32.11.5 Registers Related to Product Information

### 32.11.5.1 PRDNAMEn — Product Name Storage Register (n = 1 to 4)

This register stores the product name. The product model name is stored in 16-byte ASCII code, and PRDNAME1, PRDNAME2, PRDNAME3, and PRDNAME4 correspond to the fourth to first bytes, eighth to fifth bytes, twelfth to ninth bytes, and sixteenth to thirteenth bytes of the product model name respectively.

**Access:** This register can be read in 32-bit units.

**Address:** PRDNAME1: FFCD 00D0$_H$
PRDNAME2: FFCD 00D4$_H$
PRDNAME3: FFCD 00D8$_H$
PRDNAME4: FFCD 00DC$_H$

**Value after reset:**

| | P1H-CE (10MB) | P1H-C (8MB, CAN-FD) | P1H-C (4MB, BGA292) | P1M-C (BGA292-DPS) | P1M-C (LQFP144-DPS) |
|---|---|---|---|---|---|
| PRDNAME1 | 3746 3752$_H$ | 3746 3752$_H$ | 3746 3752$_H$ | 3746 3752$_H$ | 3746 3752$_H$ |
| PRDNAME2 | 3733 3130$_H$ | 3733 3130$_H$ | 3733 3130$_H$ | 3733 3130$_H$ | 3733 3130$_H$ |
| PRDNAME3 | 2020 4130$_H$ | 2020 2031$_H$ | 2020 2032$_H$ | 2020 2033$_H$ | 2020 2034$_H$ |
| PRDNAME4 | 2020 2020$_H$ | 2020 2020$_H$ | 2020 2020$_H$ | 2020 2020$_H$ | 2020 2020$_H$ |

| | P1H-CE (10MB, CAN-FD) | P1H-C (4MB, BGA292, CAN-FD) | P1M-C (BGA292-DPS, CAN-FD) | P1M-C (LQFP144-DPS, CAN-FD) |
|---|---|---|---|---|
| PRDNAME1 | 3746 3752$_H$ | 3746 3752$_H$ | 3746_3752$_H$ | 3746_3752$_H$ |
| PRDNAME2 | 3733 3130$_H$ | 3733 3130$_H$ | 3733_3130$_H$ | 3733_3130$_H$ |
| PRDNAME3 | 2020 4230$_H$ | 2020 4132$_H$ | 2020 4133$_H$ | 2020 4134$_H$ |
| PRDNAME4 | 2020 2020$_H$ | 2020 2020$_H$ | 2020 2020$_H$ | 2020 2020$_H$ |

| | P1H-C (4MB, BGA156, CAN-FD) | P1M-C (BGA156, CAN-FD) |
|---|---|---|
| PRDNAME1 | 3746 3752$_H$ | 3746 3752$_H$ |
| PRDNAME2 | 3933 3130$_H$ | 3933 3130$_H$ |
| PRDNAME3 | 2020 4136$_H$ | 2020 4137$_H$ |
| PRDNAME4 | 2020 2020$_H$ | 2020 2020$_H$ |

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | PRDNAMEn[31:24]*1 | | | | | | | |
| Value after reset | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | PRDNAMEn[23:16]*1 | | | | | | | |
| Value after reset | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | PRDNAMEn[15:8]*1 | | | | | | | |
| Value after reset | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PRDNAMEn[7:0]*1 | | | | | | | |
| Value after reset | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 | 0/1*2 |
| R/W | R | R | R | R | R | R | R | R |

Note 1. n = 1 to 4

Note 2. These values are depended on the product.

**Table 32.22 Contents of Product Name Storage Register**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | — | Product name fourth byte (PRDNAME1), eighth byte (PRDNAME2) twelfth byte (PRDNAME3), sixteenth byte (PRDNAME4) |
| 23 to 16 | — | Product name third byte (PRDNAME1), seventh byte (PRDNAME2) eleventh byte (PRDNAME3), fifteenth byte (PRDNAME4) |
| 15 to 8 | — | Product name second byte (PRDNAME1), sixth byte (PRDNAME2) tenth byte (PRDNAME3), fourteenth byte (PRDNAME4) |
| 7 to 0 | — | Product name first byte (PRDNAME1), fifth byte (PRDNAME2) ninth byte (PRDNAME3), thirteenth byte (PRDNAME4) |

## 32.12 Option Bytes

The flash memory has the extended area (option bytes) to store data specified by the user for various purposes. Changes in settings such as initial setting of peripheral functions using option bytes become effective after release from the reset state.

### 32.12.1 OPBT0 — Option Byte 0

**Access:** These data can only be read.

**Address:** FFCD $0030_H$

**Value after reset:** Specified by the user

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OPWD RUN0*1 | —*2 | —*2 | —*2 | OPWD OVF2*1 | OPWD OVF1*1 | OPWD OVF0*1 | OPWD0 MD*1 | OPWD VAC*1 | OPWD RUN1*1 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 |
| Value after reset | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | ETHDIS ABLE*1 | —*2 | —*2 | —*2 | —*2 | OPAUD R*1 | —*2 | —*2 | —*2 | —*2 |
| Value after reset | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. These values are depended on OPBT0 of option bytes in the flash memory that are specified by the user.

Note 2. These values are depended on OPBT0 (reserved bits) of option bytes in the flash memory that are specified by the user.

**Table 32.23 OPBT0 register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | OPWDRUN0 | This bit enables or disables an automatic start of the WDTA0.<br>0: WDTA0 automatic start is disabled.<br>1: WDTA0 automatic start is enabled. |
| 30 to 28 | Reserved | (The write value should be 1.) |
| 27 to 25 | OPWDOFV2 to OPWDOFV0 | These bits select the overflow time of the WDTA0, WDTA1. |
| 24 | OPWD0MD | This bit selects the mode of the WDTA0, WDTA1<br>0: Slow mode (WDTACLKI = 1/32 of CLK_IOSC)<br>1: Fast mode (WDTACLKI = 1/1 of CLK_IOSC) |

| OPWDOFV2 | OPWDOFV1 | OPWDOFV0 | Overflow interval time (μs) |
|---|---|---|---|
| 0 | 0 | 0 | $2^9$ / WDTACLKI |
| 0 | 0 | 1 | $2^{10}$ / WDTACLKI |
| 0 | 1 | 0 | $2^{11}$ / WDTACLKI |
| 0 | 1 | 1 | $2^{12}$ / WDTACLKI |
| 1 | 0 | 0 | $2^{13}$ / WDTACLKI |
| 1 | 0 | 1 | $2^{14}$ / WDTACLKI |
| 1 | 1 | 0 | $2^{15}$ / WDTACLKI |
| 1 | 1 | 1 | $2^{16}$ / WDTACLKI |

**Table 32.23 OPBT0 register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 23 | OPWDVAC | Selection of WDTA0, WDTA1 Variable Startup Code<br>This bit specifies the trigger register for generating a counter restart trigger to avoid counter overflow.<br>0: WDTAnWDTE (fixed)<br>1: WDTAnEVAC (variable)<br>Note: For details about WDTA start-up options, see **Section 23, Window Watchdog Timer A (WDTA)**. |
| 22 | OPWDRUN1 | This bit enables or disables an automatic start of the WDTA1 (except P1M-C).<br>[P1H-C (4MB), P1H-C (8MB), P1H-CE]<br>0: WDTA1 automatic start is disabled.<br>1: WDTA1 automatic start is enabled.<br>[P1M-C]<br>1: (The write value must be 1.) |
| 21 to 10 | Reserved | (The write value should be 1.) |
| 9 | ETHDISABLE | Ethernet Control<br>0: Ethernet module is disabled.<br>1: Ethernet module is enabled. |
| 8 to 5 | Reserved | (The write value should be 1.) |
| 4 | OPAUDR | To Select AUD RAM monitor Enable/Disable (P1H-CE only)<br>[P1H-CE]<br>0: AUDR disable<br>1: AUDR enable<br>[P1M-C, P1H-C]<br>1: (The write value must be 1.) |
| 3 to 0 | Reserved | (The write value should be 1.) |

## 32.12.2 OPBT1 — Option Byte 1

**Access:** These data can only be read.

**Address:** FFCD $0034_H$

**Value after reset:** Specified by the user

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PLL0 FREQ*1 | | —*2 | EXCLK IN*1 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 |
| Value after reset | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | —*2 | —*2 | PLL0MDIV*1 | | | PLL0NDIV*1 | | | | | | | | PLL0PDIV*1 | | |
| Value after reset | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. These values are depended on OPBT1 of option bytes in the flash memory that are specified by the user.

Note 2. These values are depended on OPBT1 (reserved bits) of option bytes in the flash memory that are specified by the user.

**Table 32.24 OPBT1 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | PLL0FREQ | Divider configuration for CLKD0DIV and CLKD1DIV and SWDT clock divider. The Bits must be set according to the PLL frequency (set by PLL0MDIV, PLL0NDIV and PLL0PDIV) and the required max. CPU frequency (CLK_CPU).<br>[P1M-C, P1H-CE]<br>00: PLL0 = 480MHz and CLK_CPU = 120MHz<br>01: PLL0 = 320MHz and CLK_CPU = 160MHz<br>10: PLL0 = 480MHz and CLK_CPU = 240MHz<br>11: prohibited setting<br>[P1H-C]<br>00: prohibited setting<br>01: PLL0 = 320MHz and CLK_CPU = 160MHz<br>10: PLL0 = 480MHz and CLK_CPU = 240MHz<br>11: prohibited setting |
| 29 | Reserved | (The write value should be 1.) |
| 28 | EXCLKIN | Selection of External Clock Instead of MOSC<br>0: Select external clock input<br>1: Select crystal |
| 27 to 14 | Reserved | (The write value should be 1.) |
| 13 to 11 | PLL0MDIV | PLL0 M-Divider Setting<br>001: 1/2 (mr = 2)<br>010: 1/3 (mr = 3)<br>other: Setting prohibited |
| 10 to 3 | PLL0NDIV | PLL0 N-Divider Setting<br>0001_1111: 1/32 (nr = 32)<br>0010_0111: 1/40 (nr = 40)<br>0010_1111: 1/48 (nr = 48)<br>0011_1011: 1/60 (nr = 60)<br>other: Setting prohibited |
| 2 to 0 | PLL0PDIV | PLL0 P-Divider Setting<br>000: 1/1 (pr = 1)<br>001: 1/2 (pr = 2)<br>other: Setting prohibited |

### 32.12.3 OPBT2 — Option Byte 2

This register contains Security information which is also described in *the RH850/P1x-C User's Manual: Hardware (Security)* ***Section 1 Basic Security***.

**Access:** These data can only be read.

**Address:** FFCD $0038_H$

**Value after reset:** Specified by the user

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | OPJTAG*1 | | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 |
| Value after reset | 0 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 |
| Value after reset | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. These values are depended on OPBT2 of option bytes in the flash memory that are specified by the user.

Note 2. These values are depended on OPBT2 (reserved bits) of option bytes in the flash memory that are specified by the user.

**Table 32.25 OPBT2 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | (The write value must be 1.) |
| 30, 29 | OPJTAG1, OPJTAG0 | Switch of the Debug Interfaces<br>The following debug interface is selected depending on the combination of the OPJTAG1 and OPJTAG0.<br>00: GPIO<br>01: LPD (4 pins)<br>10: Setting prohibited<br>11: NEXUS (JTAG) |
| 28 to 0 | Reserved | (The write value must be 1.) |

### 32.12.4 OPBT13 — Option Byte 13

This register contains Security information which is also described in *the RH850/P1x-C User's Manual: Hardware (Security)* ***Section 1 Basic Security***.

**Access:** These data can only be read.

**Address:** FFCD $0064_H$

**Value after reset:** Specified by the user

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | —*2 | —*2 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0/1 | 0/1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | CVMHDETEN*1 | CVMLDETEN*1 | —*2 | —*2 | —*2 | —*2 | —*2 |
| Value after reset | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. These values are depended on OPBT13 of option bytes in the flash memory that are specified by the user.

Note 2. These values are depended on OPBT13 (reserved bits) of option bytes in the flash memory that are specified by the user.

**Table 32.26 OPBT13 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | (The write value must be 1.) |
| 6 | CVMHDETEN | CVM High Voltage Detection Control<br>0: Disable high voltage detection<br>1: Enable high voltage detection. |
| 5 | CVMLDETEN | CVM Low Voltage Detection Control<br>0: Disable low voltage detection<br>1: Enable low voltage detection. |
| 4 to 0 | Reserved | (The write value must be 1.) |

## 32.12.5 OPBT14 — Option Byte 14

**Access:** These data can only be read.

**Address:** FFCD $0068_H$

**Value after reset:** Specified by the user

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EMF*1 | | | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | —*2 | PE2PB *1 |
| Value after reset | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | —*2 | —*2 | —*2 | —*2 | —*2 | RD_SEL*1 | | | —*2 | —*2 | —*2 | —*2 | —*2 | PE2DIS *1 | —*2 | START UPPE *1 |
| Value after reset | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. These values are depended on OPBT14 of option bytes in the flash memory that are specified by the user.

Note 2. These values are depended on OPBT14 (reserved bits) of option bytes in the flash memory that are specified by the user.

**Table 32.27 OPBT14 register contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | EMF | Emulation Mode Select Flag (P1H-CE only)<br>[P1H-CE]<br>010: P1M-C<br>011: P1H-C (4MB)<br>100: P1H-C (8MB)<br>101: P1M-C ED<br>110: P1H-C (4MB) ED<br>111: P1H-C (8MB) ED<br>Others: Setting prohibited<br>[P1M-C, P1H-C]<br>111: (The write value must be 1.) |
| 28 to 17 | Reserved | (The write value should be 1.) |
| 16 | PE2PB | Parity Bit for PE2DIS (except P1M-C)<br>[P1H-C, P1H-CE]<br>0: Parity of PE2 is disabled<br>1: Parity of PE2 is enabled<br>[P1M-C]<br>1: (The write value must be 1.) |
| 15 to 11 | Reserved | (The write value must be 1.) |
| 10 to 8 | RD_SEL | Damping resister config |
| 7 to 3 | Reserved | (The write value must be 1.) |
| 2 | PE2DIS | PE2 Disable Mode Control (except P1M-C)<br>[P1H-C, P1H-CE]<br>0: PE2 is enabled<br>1: PE2 is disabled<br>[P1M-C]<br>1: (The write value must be 1.) |

| RD_SEL | Dumping registance RD [ohm] | ΔRD (= RD-770)[ohm] |
|---|---|---|
| 000 | 1616 | 846 |
| 100 | 1402 | 632 |
| x10 | 1035 | 265 |
| xx1 | 788 | 18 |

**Table 32.27 OPBT14 register contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 1 | Reserved | (The write value must be 1.) |
| 0 | STARTUPPE | Start up PE Enabler Control<br>STARTUPPE = X & ICUMC inactive: PE1 startup<br>STARTUPPE = 0 & ICUMC active: ICUMC startup<br>STARTUPPE = 1 & ICUMC active: ICUMC startup and PE1 startup. |

## 32.12.6 OPBT15 — Option Byte 15

**Access:** These data can only be read.

**Address:** FFCD 006C$_H$

**Value after reset:** Specified by the user

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CAP_SEL*1 | | | —*2 | —*2 | AMP_SEL*1 | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1. These values are depended on OPBT15 of option bytes in the flash memory that are specified by the user.

Note 2. These values are depended on OPBT15 (reserved bits) of option bytes in the flash memory that are specified by the user.

**Table 32.28 OPBT15 register contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 7 | Reserved | (The write value must be 1.) |
| 6 to 4 | CAP_SEL | Main OSC Capacitance Configure<br>X1, 2 capacitance (Cg0, Cd0) [pF]<br>000: 0<br>001: 1<br>010: 3<br>011: 4<br>100: 5<br>101: 6<br>110: 8<br>111: 9<br>(This value is internal device capacitance. The effective capacitance has 1 to 4pF parasitic capacitance in addition to this value.) |
| 3 to 2 | Reserved | (The write value must be 1.) |
| 1, 0 | AMP_SEL | Main OSC AMP Configure<br>00: 24MHz target<br>01: 20MHz target<br>10: 16MHz target<br>11: Setting prohibited |

## 32.13 Erase Counter

For code flash, each sector of user area has 16-bit erase counter. Maximum count of erase counter is $03E9_H$ (1001), and counter is not increased if an “Erase” command is issued when erase counter is $03E9_H$.

Erase counter is located in first 704 bytes of data area. When an “Erase” command is issued to user area Flash System update erase counter (data area) first, after that code flash is erased.

**Table 32.29** and **Table 32.30** show mapping of erase counter. UNUMBER indicates block number of user area, for example, U0 indicates block 0 in **Figure 32.2**.

Erase counter is supported for code flash area of mass-produced product, and is not supported for U134 to U197 which are used only in P1H-CE (emulation device).

When valid flag equal to $5AA5\ A55A_H$, Area 1 is valid. when valid flag is other values, Area 0 is valid.

**Table 32.29 Mapping of Erase Counter (Bank A)**

| | Address (offset*1) | Bit 31 | Bit16 | Bit 15 | Bit0 |
|---|---|---|---|---|---|
| Area0 | $0000\ 0000_H$ to $0000\ 0003_H$ | Valid Flag | | | |
| | $0000\ 0004_H$ to $0000\ 003F_H$ | Reserved | | | |
| | $0000\ 0040_H$ to $0000\ 0043_H$ | Counter for U1 | | Counter for U0 | |
| | $0000\ 0044_H$ to $0000\ 0047_H$ | Counter for U3 | | Counter for U2 | |
| | $0000\ 0048_H$ to $0000\ 004B_H$ | Counter for U5 | | Counter for U4 | |
| | $0000\ 004C_H$ to $0000\ 004F_H$ | Counter for U7 | | Counter for U6 | |
| | $0000\ 0050_H$ to $0000\ 0053_H$ | Counter for U9 | | Counter for U8 | |
| | $0000\ 0054_H$ to $0000\ 0057_H$ | Counter for U11 | | Counter for U10 | |
| | | | | | |
| | $0000\ 0148_H$ to $0000\ 014B_H$ | Counter for U133 | | Counter for U132 | |
| | $0000\ 014C_H$ to $0000\ 014F_H$ | Reserved | | Reserved | |
| | $0000\ 0150_H$ to $0000\ 017F_H$ | Reserved | | | |
| Area1 | $0000\ 0180_H$ to $0000\ 02BF_H$ | Same as $0000\ 0040_H$ to $0000\ 017F_H$ (Valid area is decided by valid flag) | | | |

Note 1. Base Address of EraseCounter(Bank A) is $FF200000_H$

**Table 32.30 Mapping of Erase Counter (Bank B)**

| | Address (offset*1) | Bit 31 | Bit16 | Bit 15 | Bit0 |
|---|---|---|---|---|---|
| Area0 | $0000\ 0000_H$ to $0000\ 0003_H$ | Valid Flag | | | |
| | $0000\ 0004_H$ to $0000\ 003F_H$ | Reserved | | | |
| | $0000\ 0040_H$ to $0000\ 0043_H$ | Reserved | | Counter for U0 | |
| | $0000\ 0044_H$ to $0000\ 0047_H$ | Reserved | | Reserved | |
| | $0000\ 0048_H$ to $0000\ 004B_H$ | Reserved | | Counter for U1 | |
| | $0000\ 004C_H$ to $0000\ 004F_H$ | Reserved | | Reserved | |
| | $0000\ 0050_H$ to $0000\ 0053_H$ | Counter for U3 | | Counter for U2 | |
| | $0000\ 0054_H$ to $0000\ 0057_H$ | Counter for U5 | | Counter for U4 | |
| | | | | | |
| | $0000\ 0148_H$ to $0000\ 014B_H$ | Counter for U127 | | Counter for U126 | |
| | $0000\ 014C_H$ to $0000\ 017F_H$ | Reserved | | | |
| Area1 | $0000\ 0180_H$ to $0000\ 02BF_H$ | Same as $0000\ 0040_H$ to $0000\ 017F_H$ (Valid area is decided by valid flag) | | | |

Note 1. Base Address of EraseCounter(Bank B) is $FF300000_H$

## 32.14 ECC Test Area

In ECC test Area (**Figure 32.3**), the data to test ECC decoder of code flash is stored (**Table 32.31**). User can attempt intentional error injection to ECC decoder by reading data from this area.

**Table 32.31 ECC test data**

| | | Flash content | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ECC (bit) | | | | | | | | | Data (bit) | | | | | | | | | | | | | | | | | |
| Address | Pattern name | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 127 | 126 | 125 | 124 | 123 | 122 | 121 | 120 | ... | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0100 A000$_H$ | Walking-1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A010$_H$ | Walking-1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A020$_H$ | Walking-1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A030$_H$ | Walking-1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A040$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A050$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A060$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A070$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A080$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A090$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A0A0$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A0B0$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A0C0$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A0D0$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A0E0$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A0F0$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A100$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A810$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A820$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A830$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0100 A840$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0100 A850$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0100 A860$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0100 A870$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0100 A880$_H$ | Walking-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0100 A890$_H$ | ALL-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ... | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0100 A8A0$_H$ | ALL-0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A8B0$_H$ | Double bit | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0100 A8C0$_H$ | ALL-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ... | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0100 A8D0$_H$ | ALL-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ... | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| ... | ALL-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ... | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0100 AEF0$_H$ | ALL-1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ... | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Note:** yellow marked cells represent the Injected Error.

## 32.15 Usage Notes

1. Reading areas where programming or erasure was interrupted
   When programming or erasure of an area of flash memory is interrupted, the data stored in the area become undefined. To avoid undefined data that are read out becoming the source of faulty operation, take care not to fetch instructions or read data from areas where programming or erasure was interrupted.
2. Reading the code flash memory that has been erased but not yet been programming again
   Note that reading from an area of code flash memory that has been erased but not yet been programming again (i.e. that is in the non-programmed state) can lead to the detection of an ECC error and generation of the corresponding exception. Use blank checking when you need to confirm that an area is in the non-programmed state.
3. Prohibition of additional writing
   Writing to a given area twice is not possible. If you want to overwrite data in an area of flash memory after writing to the area has been completed, erase the area first.
4. Resets during programming and erasure
   In the case of a reset due to the signal on the RESETZ pin during programming and erasure, wait for at least the minimum value of RESETZ input low level width (See **Section 35.6.3, RESETZ Timing**) once the operating voltage is within the range stipulated in the electrical characteristics (See **Section 35.3.1, Supply Voltage Characteristics**) after assertion of the reset signal before releasing the device from the reset state (See **Section 35.6.2, Power Up/Down Timing**).
5. Allocation of vectors for interrupts and other exceptions during programming and erasure
   Generation of an interrupt or other exception during programming or erasure may lead to fetching of the vector from the code flash memory. If this does not satisfy the conditions for using background operation, set the address for vector fetching to an address that is not in the code flash memory.
6. Abnormal termination of programming and erasure (1)
   Even if programming/erasure ends abnormally due to reset input or power off, the programming/erasure state of the flash memory with undefined data cannot be verified or checked. Therefore, before using the area where programming/erasure has ended abnormally, erase the area again to prove that the corresponding area is completely erased.
7. Abnormal termination of programming and erasure (2)
   If programming and erasure of code flash memory are not completed normally, the lock bit for the target area may be enabled. In such cases, erase the block to erase the lock bit while the lock bit is in the disabled state.
8. Items prohibited during programming, erasure and blank checking
   Do not perform the following operations during programming, erasure and blank checking.
   - Have the operating voltage from the power supply go beyond the allowed range.
   - Update the values of FHVE15 and FHVE3.
   - Change the operating frequency of CLK_LSB.

9. Items prohibited during clock gear change
   Clock gear change can be done by setting control registers of clock controller. (CKSC0C,CLKD0DIV) .
   See **Section 12, Clock Controller** for details.
   Clock gear change is possible under the following conditions:

   – Bus Master do not operate EEP read access to Flash.
     To change clock ratio, it is recommended to follow the guides shown below (1) - (6) to prevent from violating the conditions above.

     (1) Inform to other CPUs that CPUx starts clock gear change.
     Wait for the operation-end of all Bus Masters(which access to EEP).
     Until (6), all Bus Masters must follow the conditions shown above.

     (2) CPUx: Execute DI instruction for interrupt disable.
     Until (5), CPUx must follow the conditions shown above.

     (3) CPUx: Write to CLKD0DIV / CKSC0C register by the target value for new clock setting.

     (4) CPUx: Read the following CLKD0STAT / CKSC0S to confirm that clock output now corresponds to the actual divider setting in CLKD0DIV / CKSC0C.

     (5) CPUx: Execute EI instruction for interrupt enable.

     (6) Inform to other CPUs that CPUx completes clock gear change.
     All Bus Masters can operate EEP read access to Flash.

     CPUx : CPU which executes clock gear change operation. PE1, PE2 or ICUMC (if enabled)

## 32.16 Reference documents

- PG-FP5 Flash Memory Programmer User's Manual
  http://documentation.renesas.com/doc/products/tool/doc/r20ut0008ej0600_pgfp5.pdf (English)
  http://documentation.renesas.com/doc/products/tool/doc/r20ut0008jj0600_pgfp5.pdf (Japanese)
- Renesas Flash Programmer Flash Programming Software User's Manual
  http://documentation.renesas.com/doc/products/tool/doc/r20ut0599ej0700_rfp.pdf (English)
  http://documentation.renesas.com/doc/products/tool/doc/r20ut0599jj0700_rfp.pdf (Japanese)

## 32.17 Difference among P1M-C, P1H-C and P1H-CE

- The size of code flash and data flash is different between P1M-C, P1H-C and P1H-CE shown in **Figure 32.2 Code Flash Memory Mapping** and **Figure 32.4 Data Flash Memory Mapping**.
- Product Name Storage Register 1 to 4 (PRDNAME1 to 4) are different between P1M-C, P1H-C and P1H-CE.
- Emulation Mode Select Flag (EMF) in Option Byte 9 register and Select AUD RAM monitor Enable/Disable (OPAUDR) in Option Byte 0 register are in P1H-CE only.
- Automatic start of the WDTA1 (OPWDRUN1) in Option Byte 0 register does not exist in P1M-C (**Table 32.23**).
- "PLL0FREQ = 00" setting is only available in P1M-C, P1H-CE (**Table 32.24**).
- Parity Bit for PE2DIS (PE2PB) in Option Byte 14 register does not exist in P1M-C (**Table 32.27**).
- PE2 Disable Mode Control (PE2DIS) is Option Byte 14 register does not exist in P1M-C (**Table 32.27**).

# Section 33 RAM Modules

## 33.1 Features

### 33.1.1 List of On-chip RAM

**Table 33.1** shows the list of on-chip RAMs for RH850/P1x-C Group (P1M-C/P1H-C/P1H-CE).

Table 33.1 List of On-chip RAM

| RAM List | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE |
|---|---|---|---|---|
| Instruction cache RAM (tag) | 256words × 4 | 256words × 4/core | 256words × 4/core | 256words × 4/core |
| Instruction cache RAM (data) | 4KB × 4 | 4KB × 4/core | 4KB × 4/core | 4KB × 4/core |
| Local RAM (PE1/PE2) | PE1: 128KB<br>PE2: —*3 | PE1: 64KB<br>PE2: 64KB | PE1: 64KB<br>PE2: 64KB | PE1: 128KB*1<br>PE2: 64KB |
| Global RAM (Bank A/B) | 320KB | 960KB | 960KB | 1344KB*2 |
| Emulation RAM | — | — | — | 2MB |
| FCU RAM | 4KB × 2 | 4KB × 2 | 4KB × 2 | 4KB × 2 |
| DTS RAM | 4KB | 4KB | 4KB | 4KB |
| CSIH RAM | 512B/channel | 512B/channel | 512B/channel | 512B/channel |
| FlexRay RAM | 8KB/module<br>512B × 2/module | 8KB/module<br>512B × 2/module | 8KB/module<br>512B × 2/module | 8KB/module<br>512B × 2/module |
| Ethernet RAM | 2KB × 2/channel<br>1KB × 1/channel | 2KB × 2/channel<br>1KB × 1/channel | 2KB × 2/channel<br>1KB × 1/channel | 2KB × 2/channel<br>1KB × 1/channel |
| GTM RAM | 1KB × 2 | 1KB × 4 | 1KB × 4 | 1KB × 4 |
| MCAN RAM | 8KB/channel | 8KB/channel | 8KB/channel | 8KB/channel |
| ICUMC Instruction cache RAM (tag) | 128words × 2 | 128words × 2 | 128words × 2 | 128words × 2 |
| ICUMC Instruction cache RAM (data) | 2KB × 2 | 2KB × 2 | 2KB × 2 | 2KB × 2 |
| Local RAM (ICUMC) | 40KB | 40KB | 40KB | 40KB |

Note 1. Local RAM size 128KB can be used only when EMF in OPBT14 is configured P1M-C or P1H-CE P1M-C mode. Local RAM size is 64KB in Other mode. For details of EMF in OPBT14, see **Section 32, Flash Memory**.

Note 2. Available RAM size depends on EMF in OPBT14.

Note 3. P1M-C does not support PE2.

## 33.2 Overview

### 33.2.1 Function overview

**Access:**

The CPU, DMAC/DTS, and H-Bus masters can access Local RAM (PE1/PE2) and Global RAM (Bank A/B). For details, see **Section 3, CPU System**.

**Emulation RAM:**

The specific area of the code flash can be replaced with this RAM per page. The access latency from the CPU after replacement is the same as that of the Code Flash.

**ECC:**

From functional safety point of view, all RAM, except for Instruction cache RAM and Emulation RAM, are protected by Error-Correcting Code (ECC) mechanism. Instruction cache RAM provides only Error-Detecting Code (EDC). For details, see **Section 28, Functional Safety**.

**RAM Initialization:**

To avoid long initialization phases by software, a hardware mechanism is implemented to initialize the following RAMs. This initialization includes correct setting of the related ECC bits.

- Local RAM (PE1/PE2)
- Global RAM (Bank A/B)
- DTS RAM
- CSIH RAM
- FlexRay RAM
- GTM RAM
- MCAN RAM
- Local RAM (ICUMC)

RAM initialization to 0 is executed by all reset. In Application Reset 1 and Limited Reset, RAM initialization can be disabled according to the setting in RAM Initialization Mode Control Registers. For details, see **Section 8 Reset Controller.** In Application Reset 1, a backup RAM can be kept by STAC_LM0 register setting. For the address of backup RAM, see **Section 4, Address Map**.

All of above RAM are checked by the MBIST. After that, the RAM is initialized. For details about MBIST, see **28.5, BIST**.

## 33.3 Emulation RAM (P1H-CE only)

RH850/P1x-C includes the emulation RAM to emulate the Code Flash. The Emulation RAM is placed for each bank of Code Flash. RH850/P1x-C has the 2-bank Code Flash and the 2-Mbyte Emulation RAM (32 k byte × 32 blocks per one-bank Code Flash). The Emulation RAM is available in the Code Flash Emulation Function described below.

### 33.3.1 Code Flash Emulation Function Using the Emulation RAM

Mapping to a Code Flash area enables a Code Flash to move to the Emulation RAM and it enables the Emulation RAM to emulate the Code Flash. The ROM data can be dynamically modified during execution of a user program via the Emulation RAM which has mapped to the Code Flash area. **Figure 33.1** shows the circuit around the Emulation RAM.

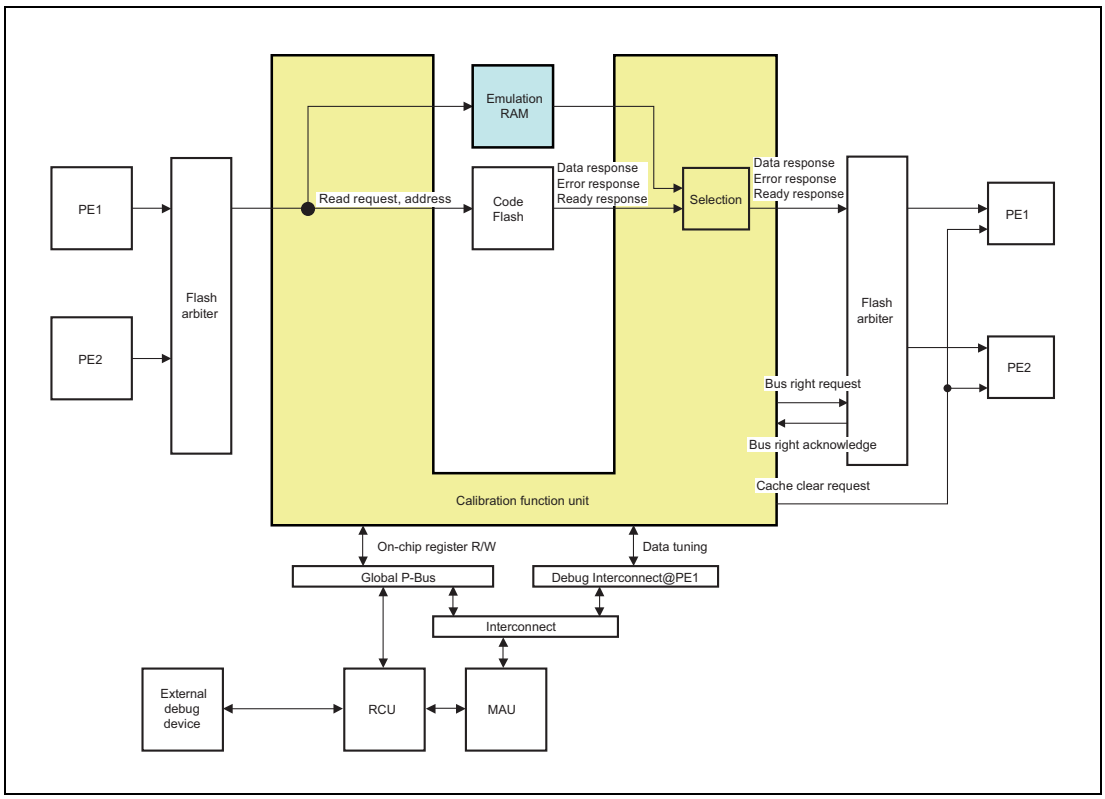



Figure 33.1 Circuit around the Emulation RAM

## 33.4 Usage Notes

When the ECC error detection/correction function is enabled for a RAM, it must be initialized with the maximum bit length of its access size before using the RAM.

If the RAM is accessed before its initialization, an ECC error may be detected. Also if initialization with the maximum bit length is not performed (e.g. if a 32-bit RAM is initialized by an 8-bit or 16-bit access), an ECC error may be detected.

On path between Local RAM and CPU, buffers are implemented to realize fast Local RAM access. Therefore when a load instruction is executed for the same address after a store instruction to Local RAM, the load instruction may read out data from buffers instead of data on Local RAM. Either of following procedures can be used to surely read data on Local RAM.

1. Read out the first written data, only after more than 32 bytes of data are written into Local RAM.
2. Execute a SYNCM instruction before a load instruction is executed on the same address after a store instruction to the Local RAM.

## 33.5 Difference among P1M-C, P1H-C and P1H-CE

See **Table 33.1** List of On-chip RAM for difference in RAM size.

The function of Emulation RAM (Section) supported only by P1H-CE.

# Section 34 Boundary Scan

## 34.1 Functional Overview

RH850/P1x-C products has JTAG Interface function conforming to IEEE 1149.1.

### 34.1.1 Features

- Five test pins (TCK, TDI, TDO, TMS and TRSTZ)
- TAP controller
- Instruction Register
- Bypass Register
- Boundary Scan Register
- BYPASS mode
- EXTEST mode
- SAMPLE/PRELOAD mode
- IDCODE mode

### 34.1.2 Block diagram

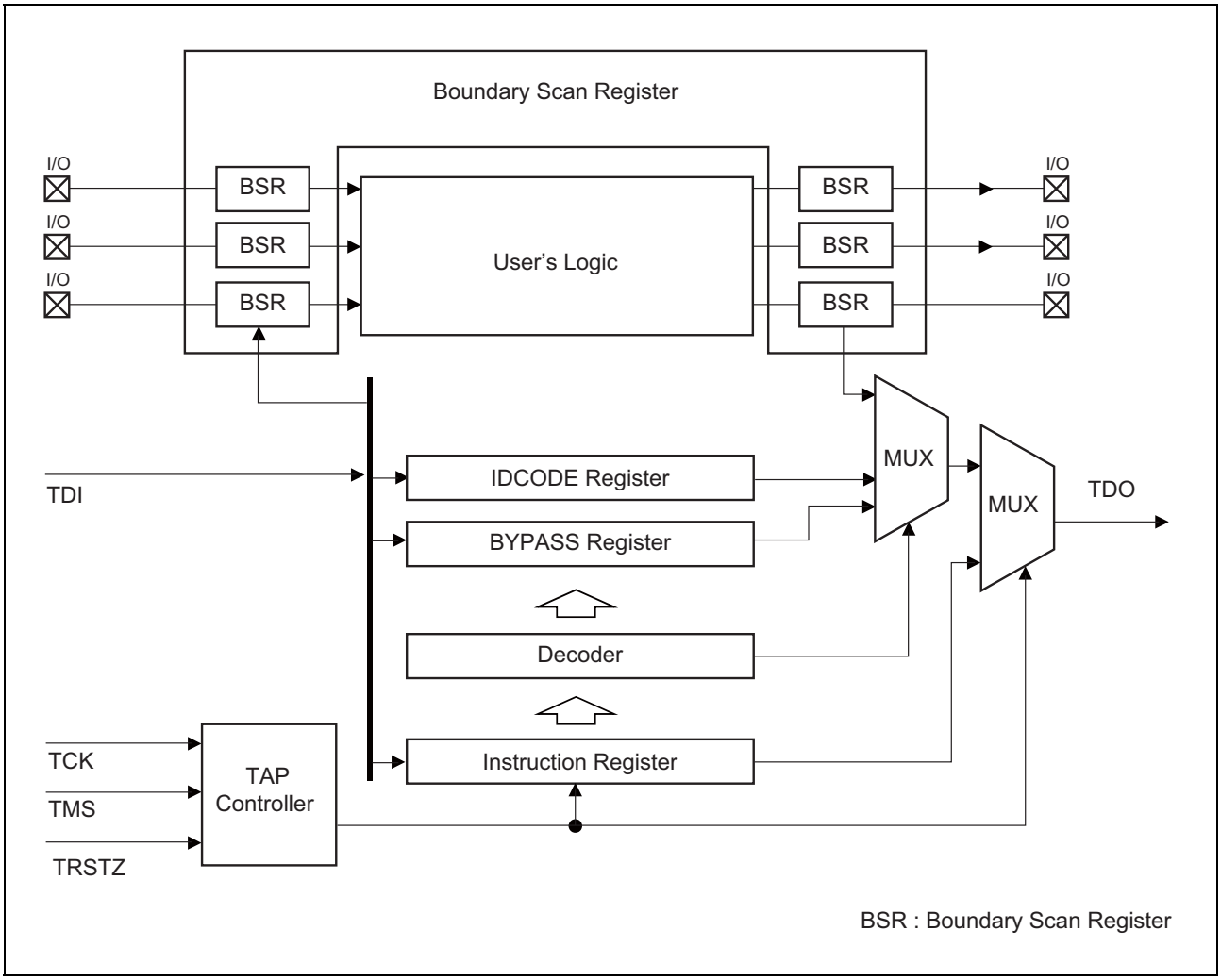



**Figure 34.1 JTAG Interface Block Diagram**

### 34.1.3 External input/output ports

Use the following pins for boundary scan I/O.

**Table 34.1 Port interface pin**

| Pin Name | I/O | Description |
|---|---|---|
| TCK | I | Test clock input pin. All input signals are captured at the rising edge of TCK, and output starts at the falling edge of TCK. |
| TDI | I | Test data input pin. It functions as the input for the serial registers placed between TDI and TDO. |
| TDO | O | Test data output pin. It functions as the output for the serial registers lined up between TDI and TDO. |
| TMS | I | Test mode selection input pin. It inputs commands to the TAP controller. |
| TRSTZ | I | Test reset input pin. This pin acknowledges asynchronous input and resets the boundary scan circuit when low-level input is received.<br>The TRSTZ pin is pulled down on the chip by using a resistor. |

### 34.1.4 Boundary Scan Mode

When performing a boundary scan, set the system to the boundary scan mode.

To switch to this mode, set up the FLMD0, FLMD1, MODE0, and MODE1 pins, and then cancel the reset.

**Table 34.2 Boundary Scan Mode Settings**

| FLMD0 | FLMD1 | MODE0 | MODE1 | Operation Mode |
|---|---|---|---|---|
| H | H | L | H | Boundary scan mode |

**Note:** For details of mode settings, refer to **Table 5.1 Mode List** of 5 Section(Operating Modes).

## 34.2 TAP Controller

### 34.2.1 Target pins

All pins are subject to boundary scan except those in **Table 34.3, Pins Not Subject to Boundary Scan**.

**Table 34.3 Pins Not Subject to Boundary Scan**

| Type | Pin Name |
|---|---|
| JTAG pins | JP0_2/TCK, JP0_0/TDI, JP0_1/TDO, JP0_3/TMS, JP0_4/TRSTZ, JP0_5/RDYZ |
| MODE pins | FLMD0, P4_5/FLMD1, P4_2/MODE0, P4_3/MODE1 |
| Analog input pins | ADC0I0 to ADC0I19, ADC1I0 to ADC1I19, A0VREFH, A1VREFH |
| Clock pins | X1, X2 |
| System pins | RESETZ, CVMOUTZ, ERROROUTZ |
| Debug pins | MSYNZ, AURORES1Z, AURORES2Z, AURORESPDZ, TODP0, TODN0, TODP1, TODN1, EVTIZ, EVTOZ, AUDRSTZ, AUDCK, AUDSYNCZ, AUDATA0 to AUDATA3, CICREFP, CICREFN, ERAMRES2Z, ERAMRESPDZ |
| Power supply pins | VDD, VCC, SYSVCC, A0VCC, A1VCC, E0VCC, E1VCC, EMUVDD, DVDD, DVCC, EMUVCC, ERAMVDD, ERAM0VCC, ERAM1VCC |
| GND pins | VSS, A0VSS, A1VSS, E0VSS, E1VSS, OSCVSS, EMUVSS, DVSS, ERAM0VSS, ERAM1VSS |

### 34.2.2 Instructions

The RH850/P1x-C product supports the BYPASS, EXTEST, SAMPLE, PRELOAD and IDCODE instructions.

**Table 34.4 Instruction Codes**

| Instructions | Instruction Codes | Remark |
|---|---|---|
| BYPASS | 1111 1111 | |
| EXTEST | 0000 0000 | |
| SAMPLE | 0100 0000 | Same code as PRELOAD |
| PRELOAD | 0100 0000 | Same code as SAMPLE |
| IDCODE | 0101 0101 | |
| | Other than the above | Setting prohibited |

### (1) BYPASS

When the BYPASS instruction is captured by the instruction register, the TAP controller enters the Shift-DR status and places the bypass register between TDI and TDO. The board-level scan path for the BYPASS instruction makes it easy to test the scan paths of other devices by using the shortest possible route.

### (2) EXTEST

The EXTEST instruction makes it possible to test an external circuit from a JTAG circuit. The test vector is specified for the boundary scan register cell on the output pin side, and the test results are captured on the input pin side. Normally, the PRELOAD instruction is executed before the EXTEST instruction to specify the first test vector for the boundary scan register cell. The result of this is that, if the TAP controller enters the Update-IR status during EXTEST instruction execution, the output driver is enabled, and the PRELOAD data is output from the output pin.

### (3) SAMPLE/PRELOAD

SAMPLE/PRELOAD is a general instruction for which the execution details are specified by IEEE 1149.1.

When the SAMPLE/PRELOAD instruction is captured by the instruction register, the TAP controller enters the Capture-DR status, and the data input to the I/O pin is captured by the boundary scan register.

### (4) IDCODE

When the IDCODE instruction is selected, IDCODE register value is output to the TDO in Shift-DR state of TAP controller. In this case, IDCODE register value is output from the LSB. During this instruction execution, test circuit does not affect the system circuit. Instruction Register is initialized by the IDCODE instruction in Test-Logic-Reset state of TAP controller.

### 34.2.3 Scan registers

#### (1) Instruction register

The instruction register is used to store instructions to be issued by the TAP controller. When the instruction register is placed between TDI and TDO, instructions are captured by the register. When the power supply is turned on, the instruction register captures an IDCODE instruction. At this time, the TAP controller is reset to the Test-Logic-Reset status.

#### (2) Bypass register

The bypass register is a 1-bit register placed between TDI and TDO. The bypass register transfers sequential test data to other devices along the shortest possible route by way of the TAP controller.

#### (3) ID register

The ID register is a 32-bit register. When the TAP controller enters the capture-DR status after the instruction register captures an IDCODE instruction, the ID register captures a 32-bit device code and manufacturer code. When the TAP controller enters the Shift-DR status, the ID register is placed between TDI and TDO.

Table 34.5 ID Register Codes

| | | ID Register Codes | | | |
|---|---|---|---|---|---|
| **Products** | **Package Name** | **31 to 28** | **27 to 12** | **11 to 1** | **0** |
| P1M-C (BGA-292) | R7F701373EABG | 0001 | 1000 0010 1100 0001 | 0100 0100 011 | 1 |
| P1M-C (BGA-292, CAN-FD) | R7F701373AEABG | 0001 | 1000 0011 0101 0100 | 0100 0100 011 | 1 |
| P1M-C (LQFP-144) | R7F701374EAFP | 0001 | 1000 0010 1100 0011 | 0100 0100 011 | 1 |
| P1M-C (LQFP-144, CAN-FD) | R7F701374AEAFP | 0001 | 1000 0011 0111 1000 | 0100 0100 011 | 1 |
| P1M-C (BGA-156, CAN-FD) | R7F701397AEABG | 0001 | 1000 0011 1100 0010 | 0100 0100 011 | 1 |
| P1H-C 4MB | R7F701372EABG | 0001 | 1000 0010 0011 1010 | 0100 0100 011 | 1 |
| P1H-C 4MB (BGA-292, CAN-FD) | R7F701372AEABG | 0001 | 1000 0011 0100 1010 | 0100 0100 011 | 1 |
| P1H-C 4MB (BGA-156, CAN-FD) | R7F701396AEABG | 0001 | 1000 0011 1100 0000 | 0100 0100 011 | 1 |
| P1H-C 8MB (CAN-FD) | R7F701371EABG | 0001 | 1000 0011 0001 0100 | 0100 0100 011 | 1 |
| P1H-CE | R7F701370AEEBG | 0001 | 1000 0010 1100 0000 | 0100 0100 011 | 1 |
| P1H-CE (CAN-FD) | R7F701370BEEBG | 0001 | 1000 0011 0100 0000 | 0100 0100 011 | 1 |

**Note:** Bit[31:28] : Revision Number. These bits indicate the revision of the product.
Bit[27:12] : Product Number. These bits indicate the product number.
Bit[11:1] : ManufactureID. These bits indicate the manufacturer number (manufacturer ID). $223_H$: Renesas Electronics.

#### (4) Boundary Scan register

The Boundary Scan register is controlled by the TAP controller. When the TAP controller enters the Capture-DR status, the boundary scan register captures the contents of the RAM I/O ring. When the TAP controller enters the Shift-DR status, the boundary scan register is placed between TDI and TDO. Several TAP instructions use the boundary scan register.

### 34.2.4 Status transitions

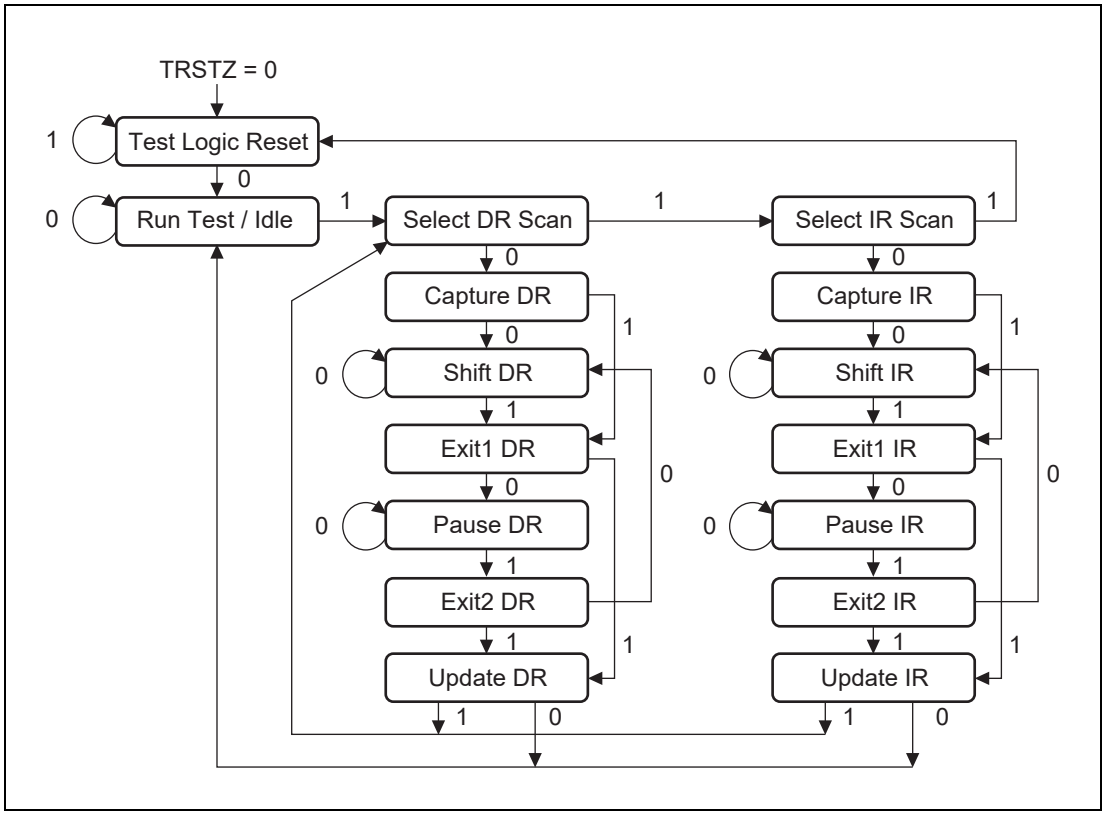



**Figure 34.2 TAP Controller Status Transitions**

## 34.3 Difference among P1M-C, P1H-C and P1H-CE

The difference among devices are Pin assignments and IDCODE.

For details, please refer to following.

**Table 34.6 Related Differences**

| Differences | Describing Section |
|---|---|
| Pin assignments | **Section 2, Pin Functions** |
| IDCODE | **Table 34.5** (ID Register Codes) |

# Section 35 Electrical Specifications

## 35.1 Overview

The specifications in this section are for devices operating under the following conditions. Where a special condition is required for a given specification, the condition will be indicated. Furthermore, the specifications in this section are not guaranteed unless the conditions listed below are met.

## 35.2 Absolute Maximum Ratings

Absolute maximum ratings are shown in the table below.

**Conditions:**

- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Power Voltage | VDD | | –0.3 | | 1.8 | V |
| | VCC | | –0.3 | | 6.5 | V |
| | SYSVCC | | –0.3 | | 6.5 | V |
| | E0VCC | | –0.3 | | 6.5 | V |
| | E1VCC | | –0.3 | | 6.5 | V |
| | A0VCC | | –0.3 | | 6.5 | V |
| | A1VCC | | –0.3 | | 6.5 | V |
| | DVCC | Emulation Device Only | –0.3 | | 4.5 | V*1 |
| | DVDD | | –0.3 | | 1.8 | V*1 |
| | EMUVCC | | –0.3 | | 4.5 | V*1 |
| | EMUVDD | | –0.3 | | 1.8 | V*1 |
| | ERAM0VCC | | –0.3 | | 4.5 | V*1 |
| | ERAM1VCC | | –0.3 | | 4.5 | V*1 |
| | ERAMVDD | | –0.3 | | 1.8 | V*1 |
| Input Voltage | VI | E0VCC pin*4 | –0.3 | | E0VCC+0.3 | V*2 |
| | | E1VCC pin*5 | –0.3 | | E1VCC+0.3 | V*2 |
| | | RESETZ X1 pins | –0.3 | | SYSVCC+0.3 | V*2 |
| | | ERAMRES2Z pin | –0.3 | | ERAM1VCC+0.3 | V*1,*2 |
| | | AURORES1Z AURORES2Z pins | –0.3 | | EMUVCC+0.3 | V*1,*2 |
| | | CICREFN CICREFP pins | –0.3 | | DVDD+0.3 | V*1,*2 |
| | | AUDR I/F, EVTIZ pins | –0.3 | | E1VCC+0.3 | V*1,*2 |
| | | ERAMRESPDZ pin | –0.3 | | E1VCC+0.3 | V*1,*2 |
| | | AURORESPDZ, MSYNZ pins | –0.3 | | E0VCC+0.3 | V*1,*2 |
| Analogue reference voltage | A0VREFH | | –0.3 | | A0VCC +0.3 | V*2 |
| | A1VREFH | | –0.3 | | A1VCC +0.3 | V*2 |

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Analogue input voltage | VIAN | A0VCC pins | –0.3 | | A0VCC +0.3 | V*2 |
| | | A1VCC pins | –0.3 | | A1VCC +0.3 | V*2 |
| Output low current *3 | $I_{OL1p}$ | per pin | | | 10 | mA |
| | $I_{OLall}$ | Sum of all output low currents of all EnVCC/AnVCC pins | | | 200 | mA |
| Output high current *3 | $I_{OH1p}$ | per pin | | | –10 | mA |
| | $I_{OHall}$ | Sum of all output high currents of all EnVCC/AnVCC pins | | | –200 | mA |
| JunctionTemperature | Tj | BGA Package | –40 | | 150 | °C |
| | | QFP Package | –40 | | 160 | |
| Storage Temperature | Tstg | BGA Package | –55 | | 150 | °C |
| | | QFP Package | –55 | | 160 | |

Note 1. Emulation Device only

Note 2. Input voltage of CICREFN and CICREFP must not exceed 1.8V, and Input voltage of other pins must not exceed 6.5V (4.5V in case of P1H-CE device).

Note 3. Given specification includes injected currents. For injected current refer to **Section 35.4.6**.

Note 4. P0, P3_11, P5, P6_2-15, P9, JP0*6, FLMD0, CVMOUTZ

Note 5. P1, P2, P3_0-10, P3_12-14, P4, P6_0-1, P7, P8, ERROROUTZ

Note 6. For P1H-CE device the JP0 is connected to E1VCC

**CAUTION**

- **Even momentarily exceeding the absolute maximum rating for just one item creates a threat of failure in the reliability of the products. That is, the absolute maximum ratings are the levels that raise a threat of physical damage to the products. Be sure to use the products only under conditions that do not exceed the ratings. The quality and normal operation of the product are guaranteed under the standards and conditions given as DC and AC characteristics.**
- **Input voltage, analog reference voltage and analog input voltage must not exceed 6.5 V**
- **Even in case when input voltage does not meet the specified characteristics, it is accepted if the injected current characteristics specified in Section 35.4.6 are met.**
- **The device's function is not guaranteed outside of the condition Section 35.3 General Characteristics.**

# 35.3 General Characteristics

## 35.3.1 Supply Voltage Characteristics

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.1 Supply Voltage Characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Core supply voltage (DPS) | VDD | DPS = dual power supply (VDD from external) | 1.20 | 1.25 | 1.35 | V*4 |
| Regulator supply voltage | VCC | | $V_{POC}$*3 | | 3.6 | V |
| System control supply voltage | SYSVCC | *1, *2 | $V_{POC}$*3 | | 3.6 | V |
| IO supply voltage | EnVCC | | $V_{POC}$*3 | | 3.6 | V |
| ADC supply voltage | AnVCC | | $V_{POC}$*3 | | 3.6 | V |
| ADC reference voltage supply | AnVREFH | | $V_{POC}$*3 | | 3.6 | V |
| Aurora I/F supply voltage (Analog) | DVCC | Emulation devices only | $V_{POC}$*3 | | 3.6 | V |
| Aurora I/F supply voltage (Digital) | DVDD | | 1.175 | 1.25 | 1.325 | V |
| ERAM IO supply voltage | ERAM1VCC | | $V_{POC}$*3 | | 3.6 | V |
| ERAM core supply voltage | ERAMVDD | | 1.20 | 1.25 | 1.35 | V |
| Aurora control IO supply voltage | EMUVCC | | $V_{POC}$*3 | | 3.6 | V |
| Aurora control core supply voltage | EMUVDD | | 1.175 | 1.25 | 1.325 | V |
| VDD slew rate @power-up | VDDsru | | | | 500 | V/ms |
| VCC slew rate @power-up | VCCsru | | | | 500 | V/ms |
| SYSVCC slew rate @power-up | SYSVCCsru | | | | 500 | V/ms |
| EnVCC slew rate @power-up | EnVCCsru | | | | 500 | V/ms |
| AnVCC slew rate @power-up | AnVCCsru | | | | 500 | V/ms |
| AnVREFH slew rate @power-up | AnVRFHsru | | | | 500 | V/ms |

Note 1. The device will operate as normal above the transistor threshold voltage level. Above this threshold level the device will operate or it is controlled by RESET/POC — depending on CVM.

Note 2. SYSVCC is monitored by POC, see chapter POC Characteristics.

Note 3. Specification value "$V_{POC}$" means: supply voltage higher than POC threshold value $V_{POC}$ or higher or equal to 3.0V. For $V_{POC}$ specification please refer to **Section 35.7.1, POC Characteristics**.

Note 4. During RESET condition an enlarged VDD up to 1.55V is accepted for a maximum time of 150s over lifetime.

## 35.3.2 Main Oscillator Characteristics

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.2 Main Oscillator Characteristics**

| Item | Symbol | Comment | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Frequency | $f_{MOSC}$ *1*2 | Crystal | | 16 –1% | | 24 + 1% | MHz |
| Stabilization time | $t_{MOST}$ | | | 3*5 | | | ms |
| Internal Capacitor size selectable by CAP_SEL setting (OPBT setting)*4 | Ccapsel | CAP_SEL [2:0] | =000b | | 0 | | pF |
| | | | =001b | | 1 | | pF |
| | | | =010b | | 3 | | pF |
| | | | =011b | | 4 | | pF |
| | | | =100b | | 5 | | pF |
| | | | =101b (default) | | 6 | | pF |
| | | | =110b | | 8 | | pF |
| | | | =111b | | 9 | | pF |
| Internal damping register size selectable by RDSEL setting (OPBT setting) | Rrdsel | RDSEL [2:0] | =000b | | 846 | | Ω |
| | | | =100b | | 632 | | Ω |
| | | | =x10b | | 265 | | Ω |
| | | | =xx1b | | 18 | | Ω |

Note 1. To reach internal usable clocks only following 3 fMOSC frequencies are supported: 16MHz, 20MHz and 24MHz. Tolerance of external quartz crystal is assumed as +/–1%.

Note 2. The StartUp is supported without external components under following conditions:
- Default values for drivability and capacitance are defined as follows
  - — Maximum drivability (AMPSEL = 00b)
  - — Default damping resistor configuration (RDSEL = xx1b)
  - — Default internal capacitance of 6pF (CAPSEL = 101b)
- Specification covers a maximum external stray capacitance of up to 6pF to each pin X1 and X2.
- After StartUp drivability and capacitance will be configured by OPBT
- A possible exceeding of the recommended maximum drive level for a crystal for all start-up phases has to be agreed with the crystal manufacturers separately.

Note 3. Depends on characteristics of selected crystal

Note 4. The CAP_SEL capacity values refer to the selectable internal device capacitances. The effective capacitance will be 1-4pF higher due to parasitic capacitances.

Note 5. Depends on characteristics of selected crystal. External reset must not be released, before oscillation becomes stable.

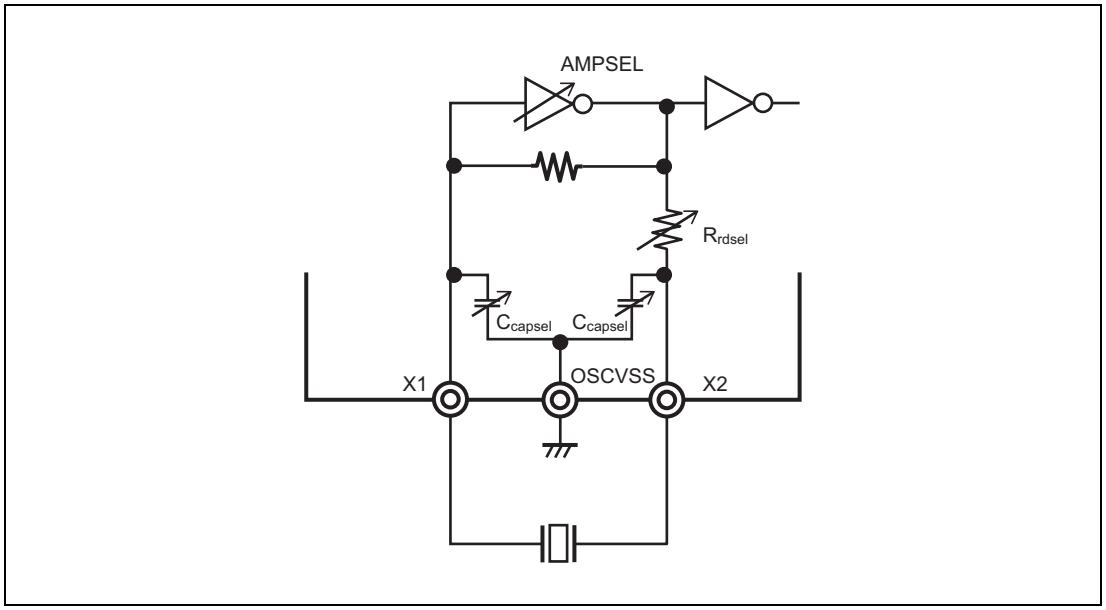



**Figure 35.1 Oscillator Circuit Diagram and Connection Example of Crystal Resonator**

**NOTE**

**Figure 35.1** shows how to connect a crystal oscillator. For crystal oscillator that are tested and recommended by Renesas (inquire separately) any external components such as a load capacitor and a dumping resistor are not necessarily required for oscillation in general. However, proper operation should be verified under actual conditions prior to use including internal setting of AMPSEL, Ccapsel and Rrdsel.

### 35.3.3 PLL0 Characteristics

**CAUTIONS**

1. **PLL0 uses output of main (crystal) oscillator as input clock.**
2. **Below specification is based on the condition of an ideal clock input.**

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.3 PLL0 Characteristics*1**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| PLL output period jitter | $t_{PJ}$ | | –200 | | +200 | ps |
| PLL output long term jitter | $t_{LTJ}$ | term = 1 µs | –500 | | +500 | ps |

Note 1. Input clock: main oscillator. Condition: Ideal clock input.

### 35.3.4 Internal Oscillator Characteristics

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.4 Internal Oscillator Characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Frequency | $f_{RH1}$ | accuracy after trimming: +/–10% | 14.4 | 16.0 | 17.6 | MHz |

### 35.3.5 Operational Condition

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.5 Operational Condition**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| CPU operating clock frequency*1 | $f_{CPUCLK}$ | | | | 240 | MHz |
| Junction temperature*2 | $T_j$ | For devices P1M-C (LQFP) | –40 | | 160*4 | °C |
| | | For devices P1M-C (BGA-292) | –40 | | 150*4 | °C |
| | | For devices P1M-C (BGA-156) | –40 | | 150*4 | °C |
| | | For devices P1H-C (4MB) | –40 | | 150*4 | °C |
| | | For devices P1H-C (8MB) | –40 | | 150*4 | °C |
| | | For devices P1H-CE | –40 | | 150*4 | °C |
| Absolute DC output current per power supply pin | $I_{O1s}$ | *3 | | | 70 | mA |
| Ext. pull-down resistor for FLMD0 | RPDMD0 | *5 | 95 | | | kΩ |
| | | *6 | 0 | | 750 | Ω |

Note 1. Please refer also to **Section 3, CPU System** and **Section 12, Clock Controller**.

Note 2. Details of thermal characteristics are described in **Section 35.8, Thermal Characteristics**.

Note 3. 1. Absolute value of the sum of DC output high and low currents of one EnVCC/AnVCC pin.
2. Regarding AC output currents, simultaneous switching of [max.] 8 output pins is allowed under the condition that the average current over lifetime does not exceed the limit given above for Item.

Note 4. In case of hotspots local Tj might exceeded. The function of the device is still ensured by test margin and library spec.

Note 5. An ext. pull-down resistor connected to FLMD0 pin is recommended in case flash programming will be used.

Note 6. A low resistance connection of FLMD0 pin to EnVSS prohibits any flash programming.

NOTES

1. If not otherwise stated, defines the valid condition for the specification values within this document.
2. The maximum frequency of a peripheral function is depending on the setting of the according clock domain in which the peripheral function resides. Please refer to **Section 12, Clock Controller**.

### 35.3.6 IO Capacitances

**Conditions:**

- Temperature range: Tjmin to Tjmax.

Table 35.6 I/O Capacitances*1*2

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input capacitance | $C_I$*3 | f = 1 MHz<br>0 V for non measurement pins | | | 10 | pF |
| Input/output capacitance | $C_{IO}$*4 | | | | 10 | pF |
| Output capacitance | $C_O$*5 | | | | 10 | pF |
| Input capacitance X1 | $C_{X1}$ | Main OSC. EXCLK mode, CAP_SEL = 010 | | | 10 | pF |

Note 1. Please note that for pins of the main oscillator the capacitance value given above are exceeded

Note 2. For analog input pins (ADCnIm) please refer to the **Section 35.7.3, A/D Converter Characteristics**

Note 3. $C_I$: capacitance of between the input pin and ground

Note 4. $C_{IO}$: capacitance of between the input/output pin and ground

Note 5. $C_O$: capacitance of between the output pin and ground

# 35.4 DC Characteristics

## 35.4.1 Input Leakage Current

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.7 Input Leakage Current**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input leakage current, high | $I_{LIH1}$ *1 | Digital I/O pins, VI = 0 to EnVCC | | | 1.0 | µA |
| Input leakage current, low | $I_{LIL1}$ *1 | Digital I/O pins, VI = 0 to EnVCC | | | –1.0 | µA |
| Input leakage current, high | $I_{LIH2}$ | RESETZ, VI = 0 to SYSVCC | | | 1.0 | µA |
| Input leakage current, low | $I_{LIL2}$ | RESETZ, VI = 0 to SYSVCC | | | –1.0 | µA |
| Input leakage current, high | $I_{LIH3}$ *2 | Analog input pins (ADCnlm), VIAN = 0 to AnVCC | | | 0.15 | µA |
| | | | | | 1.0 | µA |
| Input leakage current, low | $I_{LIL3}$ *2 | Analog input pins (ADCnlm), VIAN = 0 to AnVCC | | | –0.15 | µA |
| | | | | | –1.0 | µA |
| Input leakage current, high | $I_{LIH4}$ *3 | AUDR I/F, VIAU = 0 to EnVCC | | | 5.0 | µA |
| Input leakage current, low | $I_{LIL4}$ *3 | AUDR I/F, VIAU = 0 to EnVCC | | | –25.0 | µA |
| Input leakage current, high | $I_{LIH5}$ *1 | ERAMRES2Z, VI = 0 to ERAM1VCC | | | 1.0 | µA |
| Input leakage current, low | $I_{LIL5}$ *1 | ERAMRES2Z, VI = 0 to ERAM1VCC | | | –1.0 | µA |
| Input leakage current, high | $I_{LIH6}$ *1 | AURORESnZ, VI = 0 to EMUVCC | | | 1.0 | µA |
| Input leakage current, low | $I_{LIL6}$ *1 | AURORESnZ, VI = 0 to EMUVCC | | | –1.0 | µA |

Note 1. Pull-up/pull-down current sources/sinks are switched off.

Note 2. Leakage current under the condition that ADC conversion of the according analog input pin is switched off. When switched on, 1.0µA for high and -1.0µA for low. Both do not include charging current of the conversion sample circuits. Charging current can be judged with the equivalent input circuit and sampling time specified in **Section 35.7.3, A/D Converter Characteristics**.

Note 3. Emulation devices only

**NOTE**

For Aurora I/F there is no leakage current specification, because it is based on AC coupling only.

## 35.4.2 Digital IO Pins Input and Output Characteristics

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.8 Digital IO Pins: Input and Output Characteristics (1/4)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| High level input voltage | $V_{IH11}$ | CMOS, EnVCC = 3.0 to 3.6 V (Emulation devices only) | 0.65 × EnVCC | | EnVCC + 0.3 | V |
| | $V_{IH12}$ | SHMT1, EnVCC = 3.0 to 3.6 V | 0.65 × EnVCC | | EnVCC + 0.3 | V |
| | $V_{IH13}$ | SHMT2, EnVCC = 3.0 to 3.6 V | 0.75 × EnVCC | | EnVCC + 0.3 | V |
| | $V_{IH14}$ | SHMT4, EnVCC = 3.0 to 3.6 V | 0.80 × EnVCC | | EnVCC + 0.3 | V |
| Low level input voltage | $V_{IL11}$ | CMOS, EnVCC = 3.0 to 3.6 V (Emulation devices only) | –0.3 | | 0.35 × EnVCC | V |
| | $V_{IL12}$ | SHMT1, EnVCC = 3.0 to 3.6 V | –0.3 | | 0.35 × EnVCC | V |
| | $V_{IL13}$ | SHMT2, EnVCC = 3.0 to 3.6 V | –0.3 | | 0.25 × EnVCC | V |
| | $V_{IL14}$ | SHMT4, EnVCC = 3.0 to 3.6 V | –0.3 | | 0.50 × EnVCC | V |
| Input hysteresis for Schmitt | $V_{H11}$ | SHMT1, EnVCC = 3.0 to 3.6 V | 0.3 | | | V |
| | $V_{H12}$ | SHMT2, EnVCC = 3.0 to 3.6 V | 0.2 × EnVCC | | | V |
| | $V_{H13}$ | SHMT4, EnVCC = 3.0 to 3.6 V | 0.1 | | | V |
| Input pull-up current source *2 | $I_{PU11}$ | EnVCC = 3.0 to 3.6 V, VI = EVSS*7 | –150 | | –30 | µA |
| | | EnVCC = 3.3V, VI = EVSS*7, $T_J$ = –40°C | | –100 | | µA |
| | | EnVCC = 3.3V, VI = EVSS*7, $T_J$ = 25°C | | –80 | | µA |
| | | EnVCC = 3.3V, VI = EVSS*7, $T_J$ = 150°C | | –60 | | µA |
| Input pull-down current source*3 | $I_{PD11}$ | EnVCC = 3.0 to 3.6 V, VI = EnVCC*7 | 30 | | 150 | µA |
| | | EnVCC = 3.3V, VI = EnVCC*7, $T_J$ = –40°C | | 100 | | µA |
| | | EnVCC = 3.3V, VI = EnVCC*7, $T_J$ = 25°C | | 80 | | µA |
| | | EnVCC = 3.3V, VI = EnVCC*7, $T_J$ = 150°C | | 60 | | µA |
| Output resistance of GPIO buffer*6*8 | $R_{O11}$ | Drive strength 1, EnVCC = 3.0 to 3.6 V Forcing 0.4 V to 4 pins simultaneously or Forcing EnVCC – 0.4 V to 4 pins simultaneously | 15 | 50 | 110 | Ω |
| | $R_{O12}$ | Drive strength 2, EnVCC = 3.0 to 3.6 V Forcing 0.4 V to 8pins simultaneously or Forcing EnVCC – 0.4 V to 8 pins simultaneously | 30 | 100 | 220 | Ω |
| | $R_{O13}$ | Drive strength 3, EnVCC = 3.0 to 3.6 V Forcing 0.4 V to 16 pins simultaneously or Forcing EnVCC – 0.4 V to 16 pins simultaneously | 60 | 200 | 440 | Ω |
| | $R_{O14}$ | Drive strength 4, EnVCC = 3.0 to 3.6 V Forcing 0.4 V to 16 pins simultaneously or Forcing EnVCC – 0.4 V to 16 pins simultaneously | 120 | 400 | 880 | Ω |

Table 35.8 Digital IO Pins: Input and Output Characteristics (2/4)

| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Output resistance of HSIO buffer*5*8 | $R_{O21}$ | | Drive strength 1, E1VCC = 3.0 to 3.6 V Forcing 0.4 V to 4 pins simultaneously or Forcing E1VCC – 0.4 V to 4 pins simultaneously | 6 | 20 | 44 | Ω |
| | $R_{O22}$ | | Drive strength 2, E1VCC = 3.0 to 3.6 V Forcing 0.4 V to 4 pins simultaneously or Forcing E1VCC – 0.4 V to 4 pins simultaneously | 15 | 50 | 110 | Ω |
| | $R_{O23}$ | | Drive strength 3, E1VCC = 3.0 to 3.6 V Forcing 0.4 V to 16 pins simultaneously or Forcing E1VCC – 0.4 V to 16 pins simultaneously | 60 | 200 | 440 | Ω |
| | $R_{O24}$ | | Drive strength 4, E1VCC = 3.0 to 3.6 V Forcing 0.4 V to 16 pins simultaneously or Forcing E1VCC – 0.4 V to 16 pins simultaneously | 120 | 400 | 880 | Ω |
| High level output voltage of GPIO buffer | $V_{OH11a}$ | @ Drive strength 1 | $I_{OH11}$ ≥ –4.0 mA per pin (4 output pins)*4 | EnVCC – 0.8 | | EnVCC | V |
| | | | $I_{OH11}$ ≥ –8.0 mA per pin (1 output pin)*4 | EnVCC – 0.8 | | EnVCC | V |
| | $V_{OH11b}$ | | $I_{OH11}$ ≥ –8.0 mA per pin (2 output pins)*4 | EnVCC – 1.14 | | EnVCC | V |
| | $V_{OH11c}$ | | $I_{OH11}$ ≥ –4.0 mA per pin (Ethernet pins)*4 | E1VCC – 0.6 | | E1VCC | V |
| | $V_{OH12}$ | @ Drive strength 2 | $I_{OH12}$ ≥ –2.0 mA per pin *4 | EnVCC – 0.8 | | EnVCC | V |
| | $V_{OH13}$ | @ Drive strength 3 | $I_{OH13}$ ≥ –1.0 mA per pin *4 | EnVCC – 0.8 | | EnVCC | V |
| | $V_{OH14}$ | @ Drive strength 4 | $I_{OH14}$ ≥ –0.5mA per pin *4 | EnVCC – 0.8 | | EnVCC | V |
| High level output voltage of HSIO buffer | $V_{OH21a}$ | @ Drive strength 1 | $I_{OH21}$ ≥ –8.0 mA per pin (1 output pin)*4 | E1VCC – 0.8 | | E1VCC | V |
| | $V_{OH21b}$ | | $I_{OH21}$ ≥ –8.0 mA per pin (2 output pins)*4 | E1VCC – 1.14 | | E1VCC | V |
| | $V_{OH21c}$ | | $I_{OH21}$ ≥ –4.0 mA per pin (Ethernet pins)*4 | E1VCC – 0.6 | | E1VCC | V |
| | $V_{OH22a}$ | @ Drive strength 2 | $I_{OH22}$ ≥ –4.0 mA per pin (4 output pins)*4 | E1VCC – 0.8 | | E1VCC | V |
| | | | $I_{OH22}$ ≥ –8.0 mA per pin (1 output pin)*4 | | | | |
| | $V_{OH22b}$ | | $I_{OH22}$ ≥ –8.0 mA per pin (2 output pins)*4 | E1VCC – 1.14 | | E1VCC | V |
| | $V_{OH22c}$ | | $I_{OH22}$ ≥ –4.0 mA per pin (Ethernet pins)*4 | E1VCC – 0.6 | | E1VCC | V |
| | $V_{OH23}$ | @ Drive strength 3 | $I_{OH23}$ ≥ –1.0 mA per pin*4 | E1VCC – 0.8 | | E1VCC | V |
| | $V_{OH24}$ | @ Drive strength 4 | $I_{OH24}$ ≥ –0.5mA per pin*4 | E1VCC – 0.8 | | E1VCC | V |
| Low level output voltage for GPIO buffer | $V_{OL11a}$ | @ Drive strength 1 | $I_{OL11}$ ≤ 4.0 mA per pin (4 output pins)*4 | 0 | | 0.7 | V |
| | | | $I_{OL11}$ ≤ 8.0 mA per pin (1 output pin)*4 | | | | |
| | $V_{OL11b}$ | | $I_{OL11}$ ≤ 8.0 mA per pin (2 output pins)*4 | 0 | | 1.14 | V |
| | $V_{OL11c}$ | | $I_{OL11}$ ≤ 4.0 mA per pin (Ethernet pins)*4 | 0 | | 0.4 | V |
| | $V_{OL12}$ | @ Drive strength 2 | $I_{OL12}$ ≤ 2.0 mA per pin*4 | 0 | | 0.7 | V |
| | $V_{OL13}$ | @ Drive strength 3 | $I_{OL13}$ ≤ 1.0 mA per pin*4 | 0 | | 0.7 | V |
| | $V_{OL14}$ | @ Drive strength 4 | $I_{OL14}$ ≤ 0.5mA per pin*4 | 0 | | 0.7 | V |

**Table 35.8 Digital IO Pins: Input and Output Characteristics (3/4)**

| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Low level output voltage for HSIO buffer | $V_{OL21a}$ | @ Drive strength 1 | $I_{OL21}$ ≤ 8.0 mA per pin (1 output pins)*4 | 0 | | 0.7 | V |
| | $V_{OL21b}$ | | $I_{OL21}$ ≤ 8.0 mA per pin (2 output pins)*4 | 0 | | 1.14 | V |
| | $V_{OL21c}$ | | $I_{OL21}$ ≤ 4.0 mA per pin (Ethernet pins)*4 | 0 | | 0.4 | V |
| | $V_{OL22a}$ | @ Drive strength 2 | $I_{OL22}$ ≤ 4.0 mA per pin (4 output pins)*4 | 0 | | 0.7 | V |
| | | | $I_{OL22}$ ≤ 8.0 mA per pin (1 output pin)*4 | | | | |
| | $V_{OL22b}$ | | $I_{OL22}$ ≤ 8.0 mA per pin (2 output pins)*4 | 0 | | 1.14 | V |
| | $V_{OL22c}$ | | $I_{OL22}$ ≤ 4.0 mA per pin (Ethernet pins)*4 | 0 | | 0.4 | V |
| | $V_{OL23}$ | @ Drive strength 3 | $I_{OL23}$ ≤ 1.0 mA per pin*4 | 0 | | 0.7 | V |
| | $V_{OL24}$ | @ Drive strength 4 | $I_{OL24}$ ≤ 0.5mA per pin*4 | 0 | | 0.7 | V |
| Output rise and fall times of GPIO buffer | $t_{OR111}$/ $t_{OF111}$ | Drive strength 1 ($R_{O11}$) | Cload = 15 pF, 20% to 80% | 0.5 | | 3.6 | ns |
| | $t_{OR112}$/ $t_{OF112}$ | | Cload = 20 pF, 20% to 80% | 0.7 | | 5.0 | ns |
| | $t_{OR113}$/ $t_{OF113}$ | | Cload = 30 pF, 20% to 80% | 1.0 | | 7.0 | ns |
| | $t_{OR114}$/ $t_{OF114}$ | | Cload = 100 pF, 20% to 80% | 3.4 | | 21.0 | ns |
| | $t_{OR121}$/ $t_{OF121}$ | Drive strength 2 ($R_{O12}$) | Cload = 15 pF, 20% to 80% | 1.2 | | 7.0 | ns |
| | $t_{OR122}$/ $t_{OF122}$ | | Cload = 20 pF, 20% to 80% | 1.6 | | 9.0 | ns |
| | $t_{OR123}$/ $t_{OF123}$ | | Cload = 30 pF, 20% to 80% | 2.3 | | 13.5 | ns |
| | $t_{OR131}$/ $t_{OF131}$ | Drive strength 3 ($R_{O13}$) | Cload = 20 pF, 20% to 80% | 3.1 | | 18.0 | ns |
| | $t_{OR132}$/ $t_{OF132}$ | | Cload = 30 pF, 20% to 80% | 4.6 | | 26.0 | ns |
| | $t_{OR141}$/ $t_{OF141}$ | Drive strength 4 ($R_{O14}$) | Cload = 20 pF, 20% to 80% | 6.3 | | 35.5 | ns |
| | $t_{OR142}$/ $t_{OF142}$ | | Cload = 30 pF, 20% to 80% | 9.3 | | 55.0 | ns |

**Table 35.8 Digital IO Pins: Input and Output Characteristics (4/4)**

| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Output rise and fall times of HSIO buffer | $t_{OR211}$/ $t_{OF211}$ | Drive strength 1 ($R_{O21}$) | Cload = 15 pF, 20% to 80% | | | 2.0 | ns |
| | $t_{OR221}$/ $t_{OF221}$ | Drive strength 2 ($R_{O22}$) | Cload = 15 pF, 20% to 80% | 0.5 | | 3.6 | ns |
| | $t_{OR222}$/ $t_{OF222}$ | | Cload = 20 pF, 20% to 80% | 0.7 | | 5.0 | ns |
| | $t_{OR223}$/ $t_{OF223}$ | | Cload = 30 pF, 20% to 80% | 1.0 | | 7.0 | ns |
| | $t_{OR224}$/ $t_{OF224}$ | | Cload = 100 pF, 20% to 80% | 3.4 | | 21.0 | ns |
| | $t_{OR231}$/ $t_{OF231}$ | Drive strength 3 ($R_{O23}$) | Cload = 20 pF, 20% to 80% | 3.1 | | 18.0 | ns |
| | $t_{OR232}$/ $t_{OF232}$ | | Cload = 30 pF, 20% to 80% | 4.6 | | 26.0 | ns |
| | $t_{OR241}$/ $t_{OF241}$ | Drive strength 4 ($R_{O24}$) | Cload = 20 pF, 20% to 80% | 6.3 | | 35.5 | ns |
| | $t_{OR242}$/ $t_{OF242}$ | | Cload = 30 pF, 20% to 80% | 9.3 | | 55.0 | ns |

Note 1. Specification of parameters for IO supply voltage range EnVCC = VPOC to 3.0V is not guaranteed, but no malfunction of digital IO pins will occur.

Note 2. Pn_m, JP0_0-3, JP0_5, MSYNZ, EVTIZ
Regarding to treatment for FLMD0, please refer to **Section 35.3.5, Operational Condition** for the specification of an ext. pull-down resistor required to be connected to the FLMD0 pin, see **Table 35.5, Operational Condition**.

Note 3. Pn_m, JP0_0-5, RESETZ, ERAMRESPDZ, ERAMRES2Z, AURORES2Z, AURORES1Z, AURORESPDZ
Regarding to treatment for FLMD0, please refer to **Section 35.3.5, Operational Condition** for the specification of an ext. pull-down resistor required to be connected to the FLMD0 pin, see **Table 35.5, Operational Condition**.

Note 4. The number of pin indicates simultaneous ON. In addition, total current per 1 power supply pin of $I_{OL}/I_{OH}$ is max. ±32mA (right side or left side to 1 power supply pin is max. +16/–16mA).

Note 5. P3_9, P3_10, P3_14, P8_9, P8_10, and P8_11

Note 6. JPn_m and Pn_m excluding P3_9, P3_10, P3_14, P8_9, P8_10, and P8_11

Note 7. VI is input voltage at related input pin.

Note 8. The resistance can increase above max data in case VOH11 to VOH24 or VOL11a to VOL24 exceed the condition.

### 35.4.3 Analog Input Pins (ADCnIm) Characteristics

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.9 Analog Input Pins (ADCnIm) Characteristics*3**

| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Internal pull-up resistance*1 | $R_{PU21}$ | Analog input pins (ADCnIm), VIAN = AnVSS | | 10 | 20 | 40 | kΩ |
| | | AnVCC = 3.0V*2 | VIAN = AnVSS | | | 40 | kΩ |
| | | | VIAN = AnVCC/2 | | | 25 | |
| | | | VIAN = AnVCC | | | 16 | |
| | | AnVCC = 3.6V*2 | VIAN = AnVSS | 10 | | | kΩ |
| | | | VIAN = AnVCC/2 | 6.8 | | | |
| | | | VIAN = AnVCC | 3.6 | | | |
| Internal pull-down resistance*1 | $R_{PD21}$ | Analog input pins (ADCnIm), VIAN = AnVCC | | 10 | 20 | 40 | kΩ |
| | | AnVCC = 3.0V*2 | VIAN = AnVSS | | | 10 | kΩ |
| | | | VIAN = AnVCC/2 | | | 21 | |
| | | | VIAN = AnVCC | | | 40 | |
| | | AnVCC = 3.6V*2 | VIAN = AnVSS | 2 | | | kΩ |
| | | | VIAN = AnVCC/2 | 6 | | | |
| | | | VIAN = AnVCC | 10 | | | |

Note 1. Pull-up/down resistors are only used for diagnostic test.
PU and PD resistor data includes TDE switch resistance.
For TDE switch resistance data see **Table 35.55, ADC Characteristics**.
Note 2. A linear extrapolation between the 3 specified points is allowed.
Note 3. Specification is valid for ADCnIm pins.

### 35.4.4 Main Oscillator Input Pin (X1) Characteristics

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.10 Main Oscillator Input Pin (X1) Characteristics*1**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input voltage high level | $V_{IH21}$ | SYSVCC = $V_{POC}$ to 3.6V | 0.7 × SYSVCC | | SYSVCC | V |
| Input voltage low level | $V_{IL21}$ | | 0 | | 0.30 × SYSVCC | V |

Note 1. Only valid for EXCLK mode, (selected by flash option byte).

### 35.4.5 Aurora Interface Clock Characteristics

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.11 Aurora Interface Clock Characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Differential input voltage external AC coupled | $V_{CLK}$ *1 | | 200 | | 1600 | mV |
| Recommended external AC coupling capacitor | TRF | | 75 | 100 | 200 | nF |
| Differential input resistance | ZID | | 70 | 100 | 130 | Ω |

Note 1. Peak to peak differential input voltage.

### 35.4.6 Injected Current Characteristics

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.12 Injected Current Operating Conditions*2**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Injection current per digital input | $I_{INJ_DIN}$ | $V_{IN}$ > EnVCC, VIN < EnVSS | –2.0 | | 3.0 | mA |
| Injection current per analog input | $I_{INJ_AIN}$ | $V_{IAN}$ > AnVCC, VIAN < AnVSS | –2.0 | | 3.0 | mA |
| Injection current ERAMRES2Z | $I_{INJ_DINER}$ | $V_{IN}$ > ERAM1VCC,<br>$V_{IN}$ < ERAM1VSS | –2.0 | | 3.0 | mA |
| Injection current AURORESnZ | $I_{INJ_DINAU}$ | $V_I$ > EMUVCC,<br>$V_I$ < EMUVSS | –2.0 | | 3.0 | mA |
| Total injection current of the device | $I_{INJ_TOT}$ | *1 | — | | 50.0 | mA |

Note 1. To be considered not only at supply pin but at every point of the supply part of an IO pin.
Sum of all injected currents into all pins of the device.

Note 2. The injected current operating condition for a pin will not degrade the specified performance and functionality of other ports pins which are not affected by injected current.

# 35.5 Supply Current Characteristics

## 35.5.1 General Definition

Total current consumption is defined as follows:

$I_{TOTn} = I_{VCC} + I_{VDD} (+ I_{VCC_ED} + I_{VDD_ED}) + I_{EnVCC} = I_{INT} + I_{EnVCC}$

with

- $I_{TOTn}$: total current consumption of the device.
- $I_{VCC}$: Current consumption of high-voltage (HV) domain (e.g. POC, RVG, OSC, CVM, ADC, PLL, temperature sensor, flash charge pump and regulator, ect.)
- $I_{VDD}$: Current consumption of core domain (depending on CPU frequency)
- $I_{VCC_ED}$: Current consumption of IO buffer of emulation domain
- $I_{VDD_ED}$: Current consumption of emulation core domain
- $I_{INT}$: Current consumption of IVCC and IVDD and IVCC_ED and IVDD_ED.
- $I_{EnVCC}$: Current consumption of IO buffer

**CAUTION**

**The parameter $I_{EnVCC}$ depends on customer's application and thus need to be defined by customer in order to determine the total power dissipation of a device.**

## 35.5.2 Power Supply Currents (P1M-C)

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.13 Supply Current Consumption: Device P1M-C**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Current consumption of HV domain | $I_{VCC31}$ | VCC = SYSVCC = 3.6V | | | 28 | mA |
| Current consumption of core*1 | $I_{VDD32}$ | fCPUCLK = 240 MHz, VDD = 1.35V, TJ =150 °C | | | 282 | mA |
| | $I_{VDD33}$ | fCPUCLK = 240 MHz, VDD =1.35V,TJ =160 °C | | | 305 | mA |

Note 1. Include all cores and peripherals

Note 2. Delta current is based on the max use case as reflected in IVCC31.

### 35.5.3 Power Supply Currents (P1H-C (4MB))

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.14 Supply Current Consumption: Device P1H-C (4MB)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Current consumption of HV domain | $I_{VCC41}$ | VCC = SYSVCC = 3.6V | | | 26 | mA |
| Current consumption of core*1 | $I_{VDD42}$ | $f_{CPUCLK}$ = 240 MHz, VDD = 1.35V, $T_{JMAX}$ | | | 536 | mA |

Note 1. Include all cores and peripherals

Note 2. Delta current is based on the max use case as reflected in IVCC41.

### 35.5.4 Power Supply Currents (P1H-C (8MB))

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.15 Supply Current Consumption: Device P1H-C (8MB)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Current consumption of HV domain | $I_{VCC51}$ | VCC = SYSVCC = 3.6V | | | 30 | mA |
| Current consumption of core*1 | $I_{VDD52}$ | $f_{CPUCLK}$ = 240 MHz, VDD =1.35V,$T_{JMAX}$ | | | 546 | mA |

Note 1. Include all cores and peripherals.

Note 2. Delta current is based on the max use case as reflected in IVCC51.

### 35.5.5 Power Supply Currents (P1H-CE)

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.16 Supply Current Consumption: Device P1H-CE**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Current consumption of HV domain | $I_{VCC71}$ | VCC = SYSVCC = 3.4V, MP domain | | | 30 | mA |
| | $I_{VCC_ED71}$ | VCC = SYSVCC = 3.4V, ED domain | | | 35 | mA |
| Current consumption of core (MP domain)*1 | $I_{VDD72}$ | Dual core<br>$f_{CPUCLK}$ = 240 MHz,<br>VDD = 1.32V, $T_{JMAX}$ | | | 868 | mA |
| | $I_{VDD73}$ | Single core<br>$f_{CPUCLK}$ = 240 MHz,<br>VDD =1.32V,$T_{JMAX}$ | | | 750 | mA |
| | $I_{VDD74}$ | Single core<br>fCPUCLK = 160 MHz,<br>VDD =1.32V,$T_{JMAX}$ | | | 690 | mA |
| Current consumption of core (ED domain) | $I_{VDD_ED71,eram}$ | $f_{CPUCLK}$ = 240 MHz,<br>VDD = 1.32V,<br>ERAM domain | | | 138 | mA |
| | $I_{VDD_ED71,aur}$ | $f_{CPUCLK}$ = 240 MHz,<br>VDD = 1.32V,<br>Aurora domain | | | 87 | mA |
| Total internal current consumption | $I_{INT71}$ | $f_{CPUCLK}$ = 240 MHz,<br>VDD = 1.32V,<br>VCC = SYSVCC = 3.4 V,<br>$T_{JMAX}$ | | | 1158 | mA |

Note 1. Include all cores and peripherals

Note 2. Delta current is based on the max use case as reflected in IVCC71.

# 35.6 AC Characteristics

## 35.6.1 AC Test Conditions

### 35.6.1.1 General Conditions

Below conditions are valid for all subsequent timing specifications if not noted otherwise:

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- IO supply voltage range: EnVCC = 3.0 to 3.6 V
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

If there is no specification in particular, the specified spec is the following conditions.

- Output buffer is selected 100 Ω
- Capacitive load (Cload) connected to output pins is 30pF

**NOTE**

Even though AC characteristics correspond to the nominal frequency, a main oscillator tolerance of up to 1000ppm is considered with regard to timing characteristics for communication modules.

### 35.6.1.2 Input Measurement Points

If not stated otherwise, the below given AC timing specification is based on the measurements points as follows:

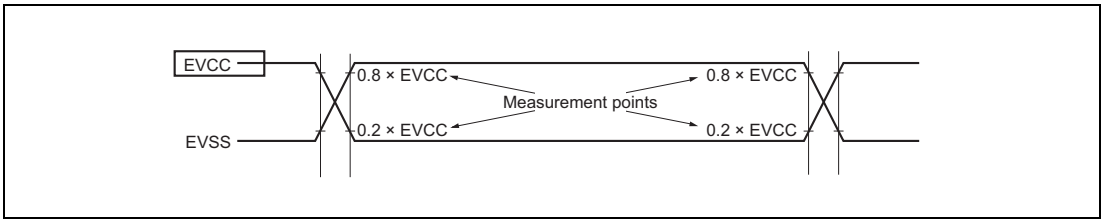



**Figure 35.2 AC Input Measurement Points Definition**

### 35.6.1.3 Output Measurement Points

If not stated otherwise, the below given AC timing specification is based on the measurements points as follows:

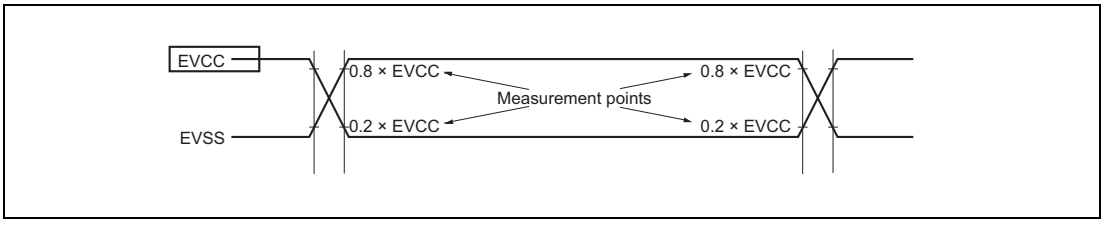



**Figure 35.3 AC Output Measurement Points Definition**

### 35.6.1.4 Load conditions

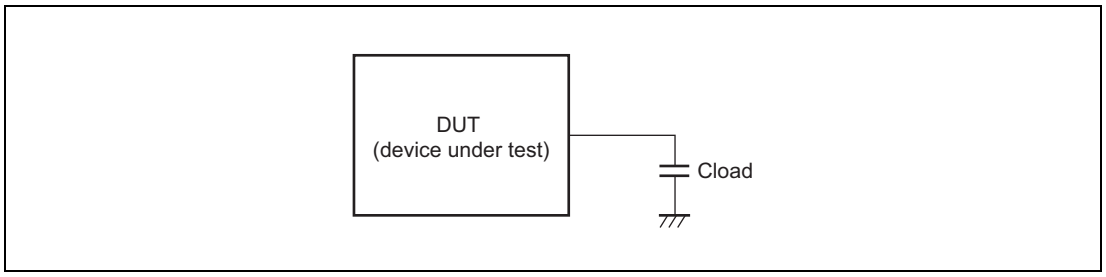



Figure 35.4 AC Load Conditions Definition

**CAUTION**

**If not otherwise stated in conditions preceding to below AC timing specifications the following capacitive load condition is valid : Cload = 30 pF.**

- **For GPIO pins with/ output drive strength 1 to 4: see the load conditions given for parameters “Output rise and fall times of GPIO buffer” in 35.4.2 Digital IO Pins Input and Output Characteristics.**
- **For HSIO pins with output drive strength 1 to 4: see the load conditions given for parameters “Output rise and fall times of HSIO buffer” in 35.4.2 Digital IO Pins Input and Output Characteristics.**

## 35.6.2 Power Up/Down Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- VCC = SYSVCC = IOVCC, with IOVCC = EnVCC, AnVCC

**Table 35.17 Power-Up/Down Timing: Using RESETZ Pin*1**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Delay time VCC ↑ & SYSVCC ↑ & IOVCC ↑ to RESETZ ↑ | $t_{DVCCPUR}$ | Power-up | $t_{MOST}$ *2 | | | ms |
| Delay time VDD ↑ to RESETZ ↑ | $t_{DVDDPUR}$ | Power-up<br>Only in case of DPS | 0 | | | ms |
| FLMD0/1 setup time time (vs RESETZ ↑) | $t_{SMDR}$ | | 1 | | | ms |
| FLMD1 hold time time (vs RESETZ ↑) | $t_{HMDR}$ | | 1 *3 | | | ms |
| MODE0/1 setup time time (vs RESETZ ↑) | $t_{SMODR}$ | | 1 | | | ms |
| MODE0/1 hold time time (vs RESETZ ↑) | $t_{HMODR}$ | | 1 | | | ms |
| Delay time RESETZ ↓ to VCC ↓ & SYSVCC↓ & IOVCC ↓ *2*4 | $t_{DRVCCPD}$ | Power-down*5 | 10.5 | | | µs |
| Delay time RESETZ↓ to VDD↓*4 | $t_{DRVDDPD}$ | Power-down*5 | 10.5 | | | µs |

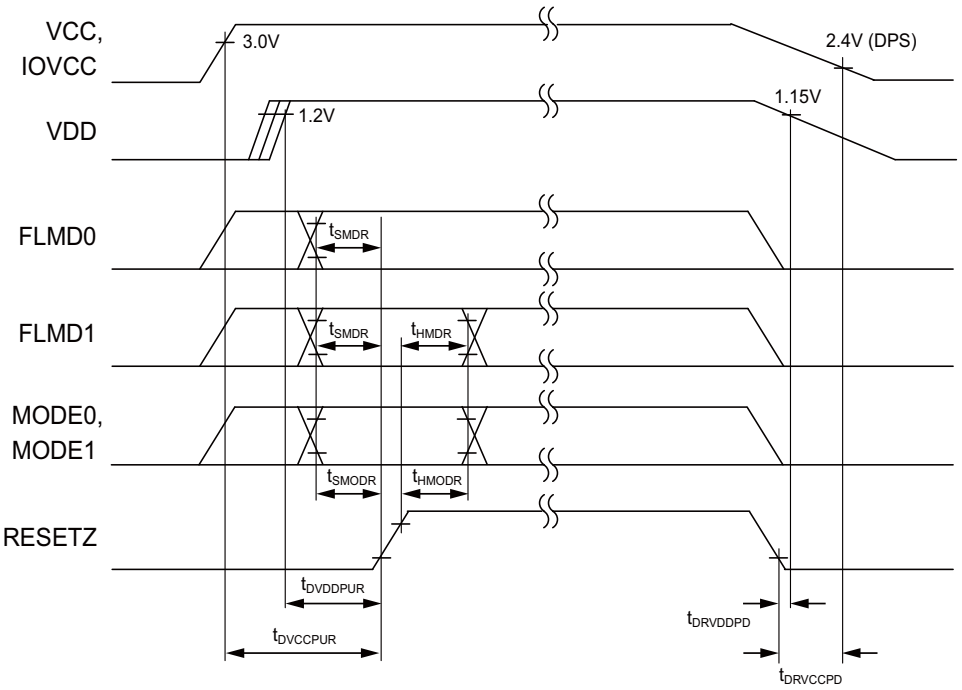
Note 1. In case power-up, power-down, RESETZ should be low. RESETZ must be asserted until CVMOUTZ is asserted and main osc. is stabilized.

Note 2. For $t_{MOST}$ refer to **Section 35.3.2, Main Oscillator Characteristics**.

Note 3. Switching of FLMD0 after RESETZ ↑ is prohibited.

Note 4. The device can withstand up to 1000 uncontrolled power down cycles without impact on lifetime. Uncontrolled means not according to power down timing requirements.

Note 5. On reset assertion the IDD and ICC will drop significantly. Consequently the dynamic behavior to VCC and VDD has to be taken in account for the delay time.




**Note:** VCC = SYSVCC = IOVCC, with IOVCC = EnVCC, AnVCC.

**Figure 35.5 Power-Up/Down Timing: Using RESETZ Pin**

### 35.6.2.1 Power Up sequencing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**

**Table 35.18 Power Up sequencing**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Runtime from RESETZ ↑ to start of instruction fetch*1 | | P1H-CE (CLK_CPU = 240MHz), P1H-C (8MB), P1H-C (4MB) and P1M-C | | | 22 | ms |
| | | P1H-CE (CLK_CPU = 120MHz and CLK_CPU = 160MHz) | | | 23 | ms |

Note 1. Including HW BIST (see **Section 8.4.5, HW BIST**), RAM initialization (see **Section 8.4.6, RAM initialization**), Read Configuration Data from FLASH (see **Section 8.4.4, Read Configuration Data from FLASH**) and PLL lock-up time (see **Section 35.3.3, PLL0 Characteristics**).

### 35.6.3 RESETZ Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**

**Table 35.19 RESETZ Timing*1**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| RESETZ input low level width*1 | $t_{WRSL}$ | *2 | 2400 | | | ns |
| RESETZ pulse rejection width*2 | $t_{WRRJ}$ | *3 | 400 | | 2400 | ns |

Note 1. The RESETZ input incorporates an analog noise filter.

Note 2. To activate the RESETZ, the low pulse width at the RESETZ input must be greater than the specified value.

Note 3. Input pulses shorter than the given min. value will be filtered out (resulting in no Reset). Input pulses between min. and max. value result in an undefined Reset condition (i.e. pulses might be filtered out or not).

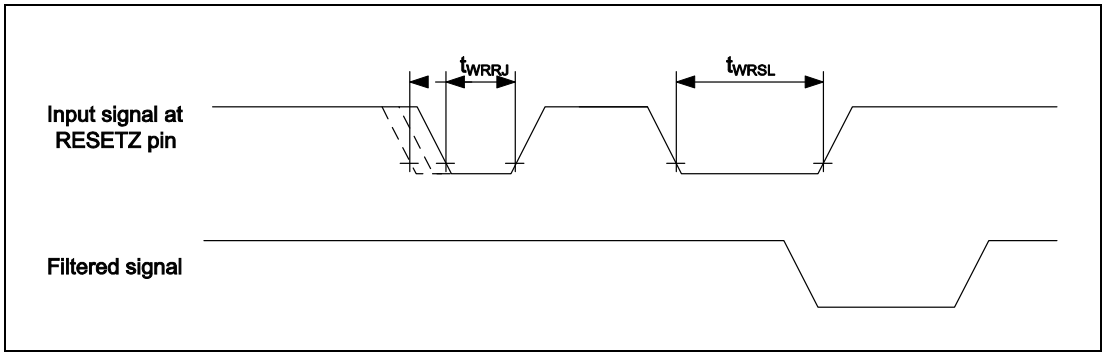



**Figure 35.6 RESETZ Timing**

## 35.6.4 Interrupt Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**

**Table 35.20 Interrupt Timing*1**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| NMI input high level width | $t_{WNIH}$ | *2 | 600 | | | ns |
| NMI input low level width | $t_{WNIL}$ | *2 | 600 | | | ns |
| NMI pulse rejection width | $t_{WNIRJ}$ | *3 | 100 | | 600 | ns |
| INTPn input high level width | $t_{WITH}$ | *2 | 600 | | | ns |
| INTPn input low level width | $t_{WITL}$ | *2 | 600 | | | ns |
| INTPn pulse rejection width | $t_{WIRJ}$ | *3 | 100 | | 600 | ns |

Note 1. Each NMI and INTPn input incorporates an analog noise filter.

Note 2. Pulses must be longer than the [min.] spec value in order to pass the filter.

Note 3. Input pulses shorter than the given min. value will be filtered out (resulting in no interrupt detected). Input pulses between min. and max. value result in an undefined interrupt signal condition (i.e. pulses might be filtered out or not).

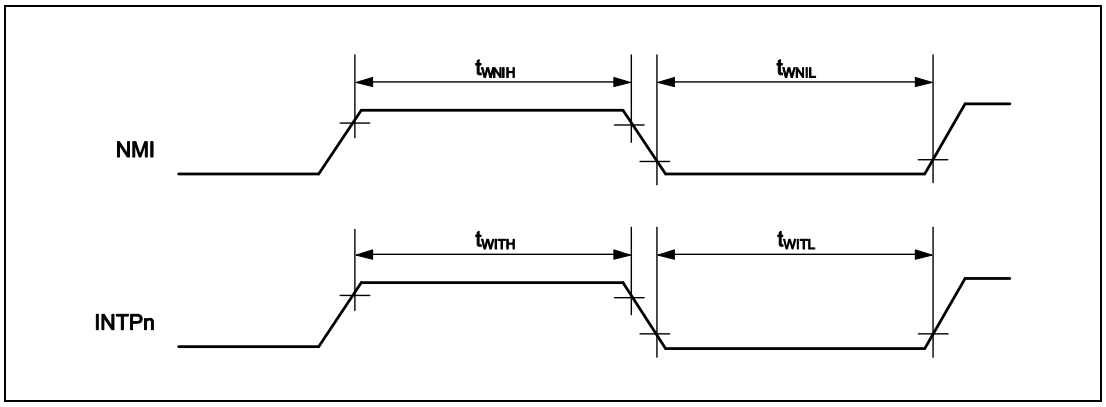



**Figure 35.7 Interrupt Timing**

### 35.6.5 Mode Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**

**Table 35.21 Mode Timing*1**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| FLMD0/1 input high level width | $t_{WFDH}$ | *2 | 600 | | | ns |
| FLMD0/1 input low level width | $t_{WFDL}$ | *2 | 600 | | | ns |
| FLMD0/1 pulse rejection width | $t_{WFDRJ}$ | *3 | 100 | | 600 | ns |

Note 1. Each FLMD0/1 input incorporates an analog noise filter.

Note 2. Pulses must be longer than the [min.] spec value in order to pass the filter.

Note 3. Input pulses between min. and max. value result in an undefined mode signal condition (i.e. pulses might be filtered out or not).This characteristic is not tested in production.

**NOTE**

Switching of FLMD0 after RESETZ ↑ is prohibited.

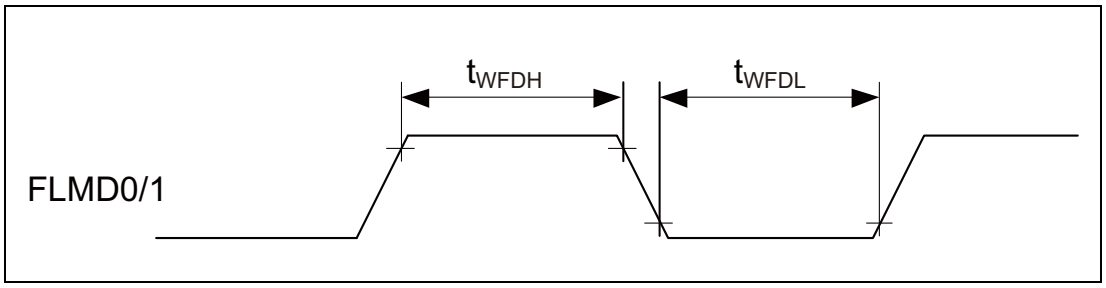



**Figure 35.8 FLMD0/1 Timing**

### 35.6.6 ADCnTRG Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**

**Table 35.22 ADCnTRG Timing*1**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| ADCnTRG input high-level width | $t_{WADH}$ | *2 | $t_{DDNF, max}$ *3 | | | ns |
| ADCnTRG input low-level width | $t_{WADL}$ | *2 | $t_{DDNF, max}$ *3 | | | ns |
| ADCnTRG pulse rejection width | $t_{WADRJ}$ | *4 | $t_{DDNFmin}$ *3 | | $t_{DDNFmax}$ *3 | ns |

Note 1. The ADC trigger (ADCnTRG) input incorporates a digital noise filter (DNF).

Note 2. Pulses must be longer than the [min.] spec value in order to pass the digital filter.

Note 3. The minimum and maximum delay time of a DNF ($t_{DDNF}$) is calculate as follows:

$$t_{DDNF,min} = (S-1) \times \frac{1}{fs}$$

$$t_{DDNF,max} = S \times \frac{1}{fs}$$

with s: number of sampling times (S = 2.5); fs: sampling clock; $f_{DNFCK}$: DNF macro clock.
Please note the $fs$ is calculated as follows:

$$fs = \frac{f_{DNFCK}}{PRS}$$ ; with PRS: prescaler (PRS = 1, 2, 4, 8, ..., 128)

Note 4. Input pulse shorter than the given min. value will be filtered out (resulting in no ADCnTRG). Input pulses between min. and max. value result in an undefined ADCnTRG signal condition (i.e. pulses might be filtered out or not).
This characteristic is not tested in production.

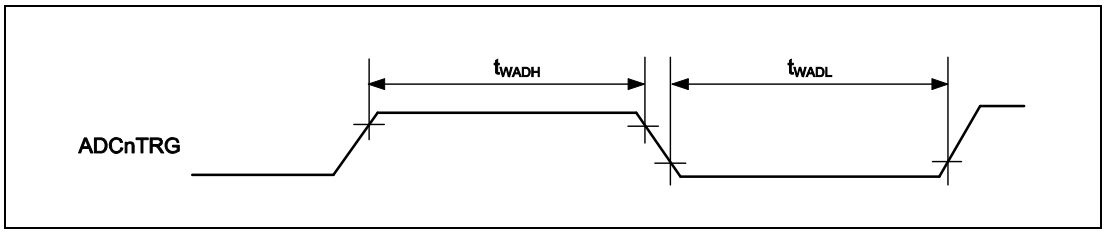



**Figure 35.9 ADCnTRG Timing**

## 35.6.7 Clock Output Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**

**Table 35.23 Clock Output Timing*1**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Clock output period time | $t_{CLKOUT}$ | | 50 | | | ns |
| Clock output high level width | $t_{WCOH}$ | | $t_{CLKOUT}$ / 2 –15 | | | ns |
| Clock output low level width | $t_{WCOL}$ | | $t_{CLKOUT}$ / 2 –15 | | | ns |

Note 1. There is a function to output the internal clock via EXTCLKnO pin.
Note 2. For base clock refer to related Section 12, Clock Controller.

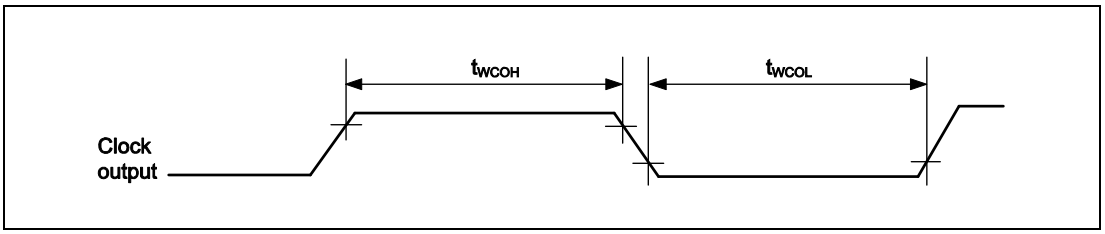



**Figure 35.10 Clock Output Timing**

## 35.6.8 High-Speed USRT Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**

**Table 35.24 HS-USRT timing (1/2)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| HSURTnCSIO to HSURTnSCKIO delay time | $t_{DCSSCI}$ | input mode (RX) | $t_{WHSCI}$ | | | ns |
| HSURTnSDIOm setup time | $t_{SSDI}$ | input mode (RX) | 7 | | | ns |
| HSURTnSDIOm hold time | $t_{SSDH}$ | input mode (RX) | 7 | | | ns |
| HSURTnSCKIO high level width | $t_{WHSCI}$ | input mode (RX) | 0.5 × $t_{KCYS}$ – 4 | | | ns |
| HSURTnSCKIO low level width | $t_{WLSCI}$ | input mode (RX) | 0.5 × $t_{KCYS}$ – 4 | | | ns |
| HSURTnSCKIO cycle time | $t_{KCYS}$ | input mode (RX) | 25 | | | ns |
| HSURTnSCKIO to HSURTnCSIO delay time | $t_{DSCICS}$ | input mode (RX) | 10 | | | ns |
| HSURTnCSIO high level width | $t_{WHCS}$ | input mode (RX) | $t_{KCYS}$ | | | ns |
| HSURTnSDIR to HSURTnCSIO delay time | $t_{DSDCSI}$ | input mode (RX) | 0 | | | ns |
| HSURTnCSIO to HSURTnSDIR delay time | $t_{DCSSDI}$ | input mode (RX) | 0 | | | ns |
| HSURTnCSIO to HSURTnSCKIO delay time | $t_{DCSSCO}$ | output mode (TX) | 0.5 × $t_{KCYM}$ | | | ns |
| HSURTnSCKIO to SDO delay time | $t_{DSCSDO}$ | output mode (TX) | 0 | | 25 | ns |
| HSURTnSDIOm hold time | $t_{HSDO}$ | output mode (TX) | 25 | | | ns |

**Table 35.24 HS-USRT timing (2/2)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| HSURTnSCKIO to HSURTnCSIO inactive time | $t_{ISCCSO}$ | output mode (TX) | 0.5 × $t_{KCYM}$ | | | ns |
| HSURTnSCKIO high level width | $t_{WHSCO}$ | output mode (TX) | 0.5 × $t_{KCYM}$ – 15 | | | ns |
| HSURTnSCKIO low level width | $t_{WLSCO}$ | output mode (TX) | 0.5 × $t_{KCYM}$ – 15 | | | ns |
| HSURTnSCKIO cycle time | $t_{KCYM}$ | output mode (TX) | 100 | | | ns |
| Interframe delay | $t_{KIFM}$ | output mode (TX) | 2 × $t_{KCYM}$ | | | ns |
| Input mode (RX) | RHSUrx | | | | 160 | Mbit/s |
| Output mode (TX) | RUSHtx | | | | 40 | Mbit/s |

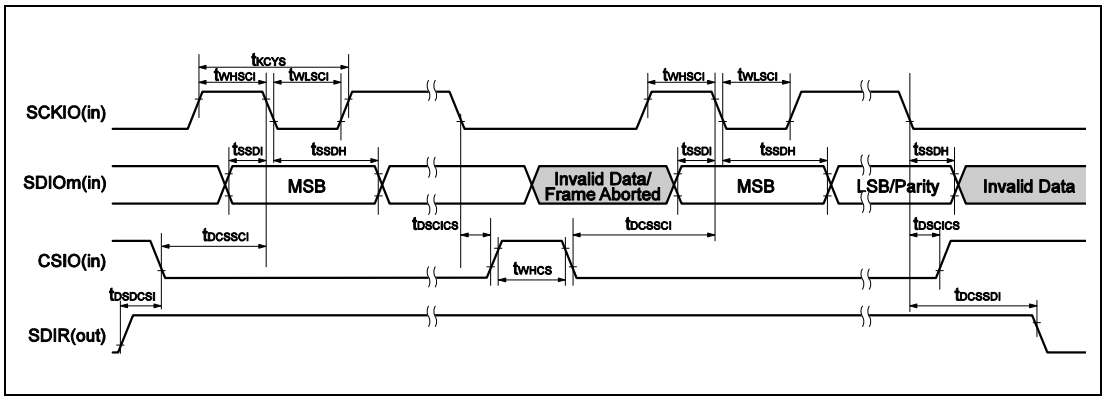



**Figure 35.11 HS-USRT Receive Timing – General definition, Aborted Frame**

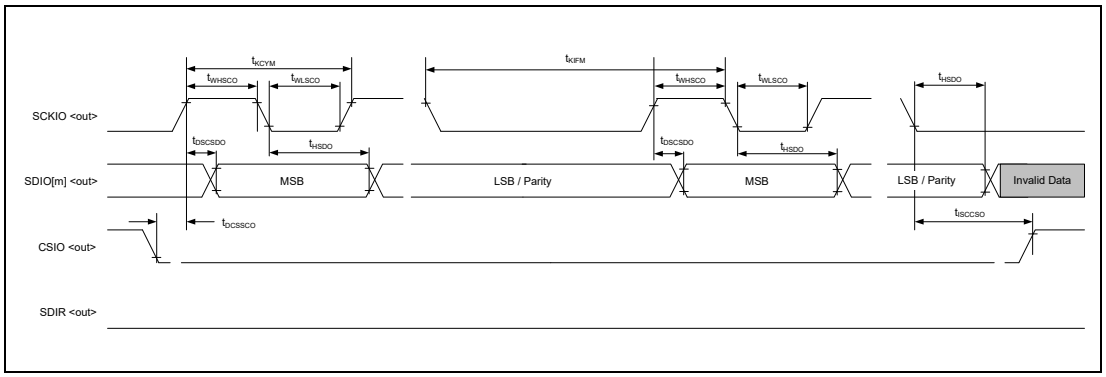



**Figure 35.12 HS-USRT Transmit Timing – General Definition, two Subsequent Frames**

## 35.6.9 CSIH Timing

### 35.6.9.1 CSIH Communication Speed Overview

Table 35.25 CSIH Communication Speed Overview

| CSIHn/pin name \ Device | | P1M-C (QFP) | P1M-C (BGA-292) | P1M-C (BGA-156) | P1H-C (4MB, BGA-292) | P1H-C (4MB, BGA-156) | P1H-C (8MB) | Port |
|---|---|---|---|---|---|---|---|---|
| CSIH0 | SC0 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P5_6 |
| | SO0 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P5_5 |
| | SI0 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P5_7 |
| | SC1 | 20MHz | 20MHz | — | 20MHz | — | 20MHz | P2_12 |
| | SO1 | 20MHz | 20MHz | — | 20MHz | — | 20MHz | P2_13 |
| | SI1 | 20MHz | 20MHz | — | 20MHz | — | 20MHz | P2_11 |
| | SC2 | 10MHz | 10MHz | — | 10MHz | — | 10MHz | P0_8 |
| | SO2 | 10MHz | 10MHz | — | 10MHz | — | 10MHz | P0_9 |
| | SI2 | 10MHz | 10MHz | — | 10MHz | — | 10MHz | P0_7 |
| CSIH1 | SC0 | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | P4_3 |
| | SO0 | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | P4_2 |
| | SI0 | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | P4_4 |
| | SC1 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P2_2 |
| | SO1 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P1_2 |
| | SI1 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P1_1 |
| | SC2 | 10MHz | 10MHz | — | 10MHz | — | 10MHz | P0_8 |
| | SO2 | 10MHz | 10MHz | — | 10MHz | — | 10MHz | P0_7 |
| | SI2 | 10MHz | 10MHz | — | 10MHz | — | 10MHz | P0_9 |
| CSIH2 | SC0 | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | P2_0 |
| | SO0 | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | P2_1 |
| | SI0 | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | P2_3 |
| | SC1 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P5_7 |
| | SO1 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P5_6 |
| | SI1 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P5_10 |
| | SC2 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P4_11 |
| | SO2 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P4_10 |
| | SI2 | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | 10MHz | P4_12 |
| CSIH3 | SC0 | 10MHz | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | P1_2 |
| | SO0 | 10MHz | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | P1_4 |
| | SI0 | 10MHz | 20MHz | 20MHz | 20MHz | 20MHz | 20MHz | P1_3 |
| | SC1 | — | 10MHz | — | 10MHz | — | 10MHz | P6_10 |
| | SO1 | — | 10MHz | — | 10MHz | — | 10MHz | P6_13 |
| | SI1 | — | 10MHz | — | 10MHz | — | 10MHz | P6_12 |
| | SC2 | — | — | — | 10MHz | — | 10MHz | P8_3 |
| | SO2 | — | — | — | 10MHz | — | 10MHz | P8_0 |
| | SI2 | — | — | — | 10MHz | — | 10MHz | P8_1 |

**Note:** Description for CSIH communication is based on CSIH-pin group. A CSIH-pin group is defined by naming having same suffix n=> e.g. CSIH1SCn/CSIH1SOn/CSIH1SIn).

### 35.6.9.2 Master mode (10 MHz Communication Speed)

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- If not other mentioned the following conditions are different from the general ones given above:
  - Capacitive load (Cload) connected to output pins: CSIHnSC = 100pF, CSIHnSO = 100pF, CSIHnCSS = 100pF.
    It is necessary to select 50 Ω output buffer.
  - Capacitive load (Cload) connected to output pins: Up to 30pF.
    Output buffer can be selected 50/100/200 Ω.
  - EnVCC = 3.0 to 3.6V
- Usable pin for 10 MHz mode is random for all CSIH-pin group within one CSIHn (same suffix n). For CSIH-pin group see **Table 35.25, CSIH Communication Speed Overview**.
- Below terminals have no "100 Ω" output buffer.
  P3_9 (CSIH1CSS4), P3_10 (CSIH1CSS5)
  So using pin Group3 of CSIH1, please select "50 Ω" or "200 Ω" output buffer about all pin Group3 of CSIH1. About Group3 of CSIH1, see "**Section 2, Pin Functions**".

**Table 35.26 CSIH Timing (Master Mode, 10 MHz Communication Speed) (1/2)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| CSIH Operation clock cycle time | $t_{KCYn}$ | | 6.25 | | | ns |
| CSIHnSC cycle time | $t_{KCYMn}$ | | 100 | | | ns |
| CSIHnSC high level width | $t_{KWHMn}$ | Cload = 20pF@50Ω output buffer select | $0.5 \times t_{KCYMn} - 7$ | | | ns |
| | | Cload = 30pF@50Ω output buffer select | $0.5 \times t_{KCYMn} - 9$ | | | ns |
| | | Cload = 100pF@50Ω output buffer select | $0.5 \times t_{KCYMn} - 23$ | | | ns |
| | | Cload = 15pF@100Ω output buffer select | $0.5 \times t_{KCYMn} - 9$ | | | ns |
| | | Cload = 30pF@100Ω output buffer select | $0.5 \times t_{KCYMn} - 15.5$ | | | ns |
| | | Cload = 20pF@200Ω output buffer select | $0.5 \times t_{KCYMn} - 20$ | | | ns |
| | | Cload = 30pF@200Ω output buffer select | $0.5 \times t_{KCYMn} - 28$ | | | ns |

**Table 35.26 CSIH Timing (Master Mode, 10 MHz Communication Speed) (2/2)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| CSIHnSC low level width | $t_{KWLMn}$ | Cload = 20pF@50Ω output buffer select | 0.5 × $t_{KCYMn}$ – 7 | | | ns |
| | | Cload = 30pF@50Ω output buffer select | 0.5 × $t_{KCYMn}$ – 9 | | | ns |
| | | Cload = 100pF@50Ω output buffer select | 0.5 × $t_{KCYMn}$ – 23 | | | ns |
| | | Cload = 15pF@100Ω output buffer select | 0.5 × $t_{KCYMn}$ – 9 | | | ns |
| | | Cload = 30pF@100Ω output buffer select | 0.5 × $t_{KCYMn}$ – 15.5 | | | ns |
| | | Cload = 20pF@200Ω output buffer select | 0.5 × $t_{KCYMn}$ – 20 | | | ns |
| | | Cload = 30pF@200Ω output buffer select | 0.5 × $t_{KCYMn}$ – 28 | | | ns |
| CSIHnSI setup time (vs. CSIHnSC)*1 | $t_{SSIMn}$ | *4 | 28 | | | ns |
| CSIHnSI hold time (vs. CSIHnSC)*1 | $t_{HSIMn}$ | *4 | 0 | | | ns |
| CSIHnSO output delay | $t_{DSOMn}$*5 | | | | 7 | ns |
| CSIHnCSS[7:0] inactive width | $t_{WSCSBn}$ | | CSIDLE × $t_{KCYMn}$ – 28 *1 | | | ns |
| CSIHnCSS[7:0] setup time (vs. CSIHnSC) | $t_{SSCSBn0}$ | CSIHnCTL1.CSIHnDAP = 0 | CSSETUP × $t_{KCYMn}$ – 10 *2 | | | ns |
| | $t_{SSCSBn1}$ | CSIHnCTL1.CSIHnDAP = 1 | (CSSETUP + 0.5) × $t_{KCYMn}$ – 10*2 | | | ns |
| CSIHnCSS[7:0] hold time (vs. CSIHnSC) | $t_{HSCSBn0}$ | CSIHnCTL1.CSIHnSIT = 0 | CSSHOLD × $t_{KCYMn}$ – 5*3 | | | ns |
| | $t_{HSCSBn1}$ | CSIHnCTL1.CSIHnSIT = 1 | (CSSHOLD + 0.5) × $t_{KCYMn}$ – 5*3 | | | ns |
| CSIHnRYI setup time (vs. CSIHnSC) | $t_{SRYI}$ | | 2 × $t_{KCYn}$ + 54 | | | ns |
| CSIHnRYI high level width | $t_{WRYI}$ | | $t_{KCYn}$ + 5 | | | ns |

Note 1. CSIDLE: Setting value of CSIHnCFGx.CSIHnIDx[2:0]

Note 2. CSSETUP: Setting value of CSIHnCFGx.CSIHnSPx[3:0]

Note 3. CSSHOLD: Setting value of CSIHnCFGx.CSIHnHDx[3:0]

Note 4. Please set SLRS = 1 if this specification can not be achieved by setting SLRS = 0 (See **Figure 35.13** and **Figure 35.14**). SLRS: Setting value of CSIHnCTL1.CSIHnSLRS

Note 5. tDSOMn timing is only valid for CSIHnSC and CSIHnSO have same output rise and fall times. If not identical the timing changes according the differences of Output rise and fall times. For tOR/tOF refer to “Output rise and fall times of GPIO buffer” in **Section 35.4.2, Digital IO Pins Input and Output Characteristics**.

### 35.6.9.3 Master mode (20 MHz Communication Speed)

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- If not other mentioned the following conditions are different from the general ones given above:
  - Capacitive load (Cload) connected to output pins: CSIHnSC = 15pF, CSIHnSO = 15pF, CSIHnCSS = 15pF.
    It is necessary to select 50 Ω output buffer.
  - EnVCC = 3.0 to 3.6V
- Usable pin for 20 MHz mode belong to same CSIH-pin group. For 20 MHz CSIH-pin group refer to **Table 35.25, CSIH Communication Speed Overview**.

**Table 35.27 CSIH Timing (Master Mode, SLRS = 1, 20 MHz Communication Speed)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| CSIH Operation clock cycle time | $t_{KCYn}$ | | 6.25 | | | ns |
| CSIHnSC cycle time | $t_{KCYMn}$ | | 50 | | | ns |
| CSIHnSC high level width | $t_{KWHMn}$ | | $0.5 \times t_{KCYMn} - 6$ | | | ns |
| CSIHnSC low level width | $t_{KWLMn}$ | | $0.5 \times t_{KCYMn} - 6$ | | | ns |
| CSIHnSI setup time (vs. CSIHnSC) | $t_{SSIMn}$ | SLRS = 1*4 | 20 | | | ns |
| CSIHnSI hold time (vs. CSIHnSC) | $t_{HSIMn}$ | SLRS = 1*4 | 0 | | | ns |
| CSIHnSO output delay (vs. CSIHnSC) | $t_{DSOMn}$ | | | | 7 | ns |
| CSIHnCSS[7:0] inactive width | $t_{WSCSBn}$ | | CSIDLE × $t_{KCYMn}$ – 23*1 | | | ns |
| CSIHnCSS[7:0] setup time (vs. CSIHnSC) | $t_{SSCSBn0}$ | CSIHnCTL1.CSIHnDAP = 0 | CSSETUP × $t_{KCYMn}$ – 10*2 | | | ns |
| | $t_{SSCSBn1}$ | CSIHnCTL1.CSIHnDAP = 1 | (CSSETUP + 0.5) × $t_{KCYMn}$ – 10*2 | | | ns |
| CSIHnCSS[7:0] hold time (vs. CSIHnSC) | $t_{HSCSBn0}$ | CSIHnCTL1.CSIHnSIT = 0 | CSSHOLD × $t_{KCYMn}$ – 5*3 | | | ns |
| | $t_{HSCSBn1}$ | CSIHnCTL1.CSIHnSIT = 1 | (CSSHOLD + 0.5) × $t_{KCYMn}$ – 5*3 | | | ns |
| CSIHnRYI setup time (vs. CSIHnSC) | $t_{SRYI}$ | | $2 \times t_{KCYn} + 48$ | | | ns |
| CSIHnRYI high level width | $t_{WRYI}$ | | $t_{KCYn} + 5$ | | | ns |

Note 1. CSIDLE: Setting value of CSIHnCFGx.CSIHnIDx[2:0]
Note 2. CSSETUP: Setting value of CSIHnCFGx.CSIHnSPx[3:0]
Note 3. CSSHOLD: Setting value of CSIHnCFGx.CSIHnHDx[3:0]
Note 4. SLRS: Setting value of CSIHnCTL1.CSIHnSLRS

### 35.6.9.4 Slave mode (10 MHz Communication Speed)

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- If not other mentioned the following conditions are different from the general ones given above:
  - Capacitive load (Cload) connected to output pins: Up to 100pF
    It is necessary to select 50 Ω output buffer.
  - Capacitive load (Cload) connected to output pins: Up to 30pF.
    Output buffer can be selected 50/100/200 Ω.
  - EnVCC = 3.0 to 3.6V
- Usable pin for 10 MHz mode is random for all CSIH-pin group within one CSIHn (same suffix n).
  For CSIH-pin group see **Table 35.25, CSIH Communication Speed Overview**.

**Table 35.28 CSIH Timing (Slave Mode, 10 MHz Communication Speed)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| CSIH Operation clock cycle time | $t_{KCYn}$ | | 6.25 | | | ns |
| CSIHnSC cycle time | $t_{KCYSn}$ | | 100 | | | ns |
| CSIHnSC high level width | $t_{KWHSn}$ | | 0.5 × $t_{KCYSn}$ – 28 | | | ns |
| CSIHnSC low level width | $t_{KWLSn}$ | | 0.5 × $t_{KCYSn}$ – 28 | | | ns |
| CSIHnSI setup time (vs. CSIHnSC) | $t_{SSISn}$ | | 10 | | | ns |
| CSIHnSI hold time (vs. CSIHnSC) | $t_{HSISn}$ | | $t_{KCYn}$ + 5 | | | ns |
| CSIHnSO output delay (vs. CSIHnSC) | $t_{DSOSn}$ | | | | 54 | ns |
| CSIHnRYO output delay | $t_{SRYOn}$ | $t_{KCYSn}$ / $t_{KCYn}$ is 8 and over | | | 54 | ns |
| | | $t_{KCYSn}$ / $t_{KCYn}$ is less than 8 | | | 54 + tKCYSn | ns |
| CSIHnSSIZ setup time (vs. CSIHnSC) | $t_{SSSIn}$ | | 0.5 × $t_{KCYSn}$ – 10 | | | ns |
| CSIHnSSIZ hold time (vs. CSIHnSC) | $t_{HSSIn}$ | | $t_{KCYn}$ + 5 | | | ns |

### 35.6.9.5 Slave mode (20 MHz Communication Speed)

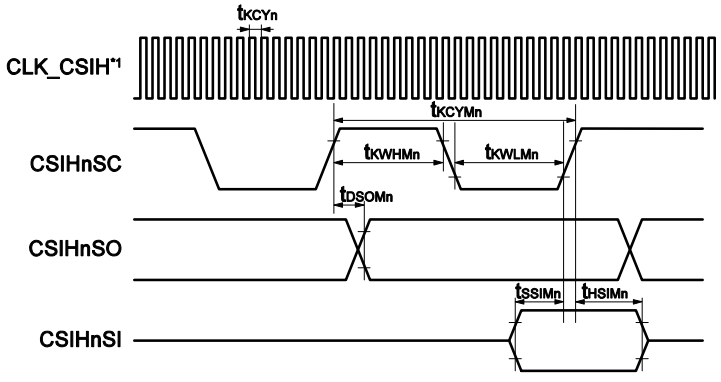
**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- If not other mentioned the following conditions are different from the general ones given above:
  - Capacitive load (Cload) connected to output pins: Up to 15pF
    It is necessary to select 50 Ω output buffer.
  - EnVCC = 3.0 to 3.6V
- Usable pin for 20 MHz mode belong to same CSIH-pin group. For 20 MHz CSIH-pin group refer to **Table 35.25, CSIH Communication Speed Overview**.

**Table 35.29 CSIH Timing (Slave Mode, 20 MHz Communication Speed)**

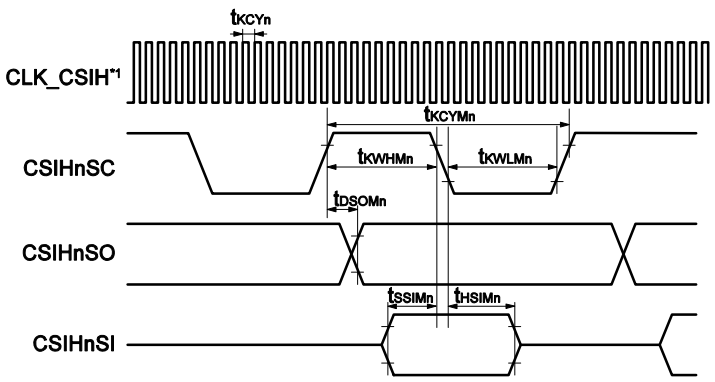
| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| CSIH Operation clock cycle time | $t_{KCYn}$ | | 6.25 | | | ns |
| CSIHnSC cycle time | $t_{KCYSn}$ | | 50 | | | ns |
| CSIHnSC high level width | $t_{KWHSn}$ | | 0.5 × $t_{KCYSn}$ – 6 | | | ns |
| CSIHnSC low level width | $t_{KWLSn}$ | | 0.5 × $t_{KCYSn}$ – 6 | | | ns |
| CSIHnSI setup time (vs. CSIHnSC) | $t_{SSISn}$ | | 10 | | | ns |
| CSIHnSI hold time (vs. CSIHnSC) | $t_{HSISn}$ | | $t_{KCYn}$ + 5 | | | ns |
| CSIHnSO output delay (vs. CSIHnSC) | $t_{DSOSn}$ | | | | 24 | ns |
| CSIHnRYO output delay | $t_{SRYOn}$ | | | | 24 | ns |
| CSIHnSSIZ setup time (vs. CSIHnSC) | $t_{SSSIn}$ | | 0.5 × $t_{KCYSn}$ – 10 | | | ns |
| CSIHnSSIZ hold time (vs. CSIHnSC) | $t_{HSSIn}$ | | $t_{KCYn}$ + 5 | | | ns |

#### (1) CSIH Waveform (Master Mode)




**Note 1.** CLK_CSIH is internal clock.

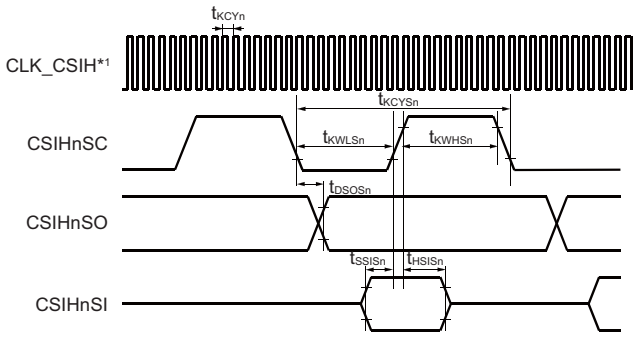
**Figure 35.13 CSIH Master Mode Timing – SLRS = 1**




**Note 1.** CLK_CSIH is internal clock.

**Note:** CSIHnCKR = 0, CSIHnCKP0 = 0, CSIHnDAP0 = 1 or CSIHnCKR = 0, CSIHnCKP0 = 1, CSIHnDAP0 = 0 or CSIHnCKR = 1, CSIHnCKP0 = 0, CSIHnDAP0 = 0

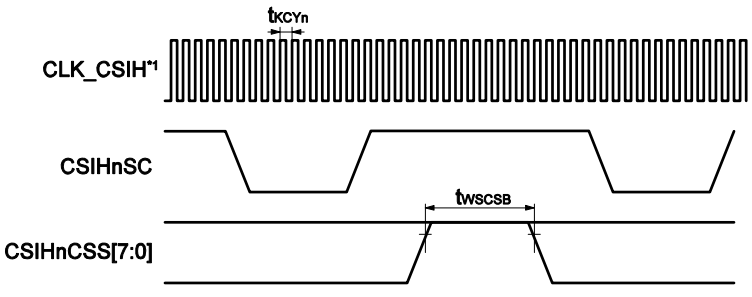
**Figure 35.14 CSIH Master Mode Timing – SLRS = 0**




**Note 1.** CLK_CSIH is internal clock.

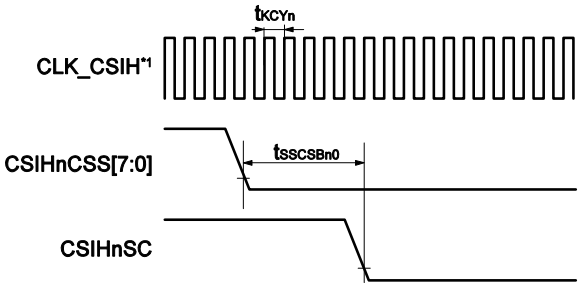
**Note:** CSIHnCKR = 0, CSIHnCKP0 = 0, CSIHnDAP0 = 0 or CSIHnCKR = 0, CSIHnCKP0 = 1, CSIHnDAP0 = 1 or CSIHnCKR = 1, CSIHnCKP0 = 0, CSIHnDAP0 = 1

**Figure 35.15 CSIH Master Mode Timing – SLRS = 0**




**Note 1.** CLK_CSIH is internal clock.

**Figure 35.16 CSIH Master Mode Timing – CSS[7:0] inactive width**




**Note 1.** CLK_CSIH is internal clock.

**Figure 35.17 CSIH Master Mode Timing – CSS[7:0] setup (CSIHnDAP = 0)**

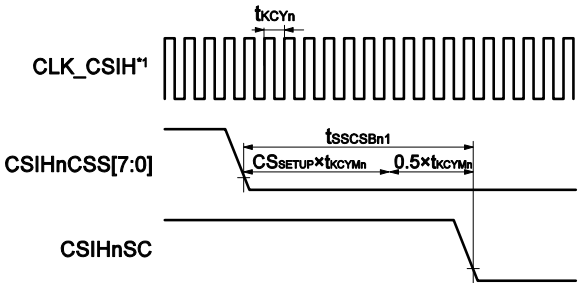
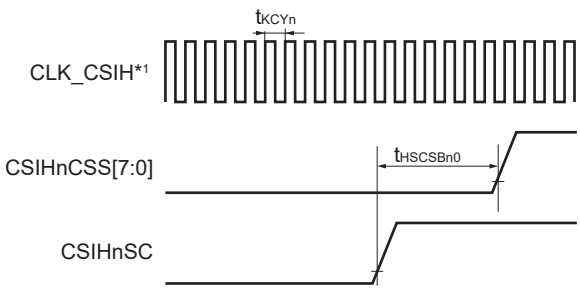
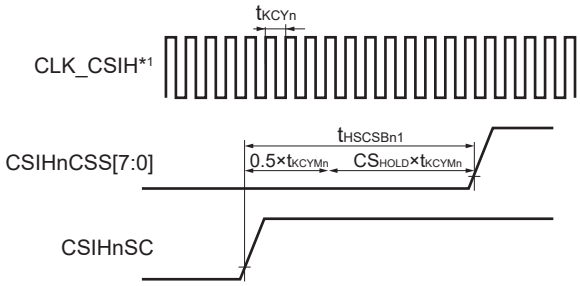



**Figure 35.18 CSIH Master Mode Timing – CSS[7:0] setup (CSIHnDAP0 = 1)**



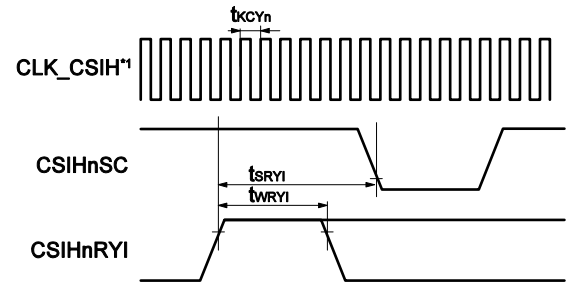

**Note 1.** CLK_CSIH is internal clock.

**Figure 35.19 CSIH Master Mode Timing – CSS[7:0] hold (CSIHnSIT = 0)**



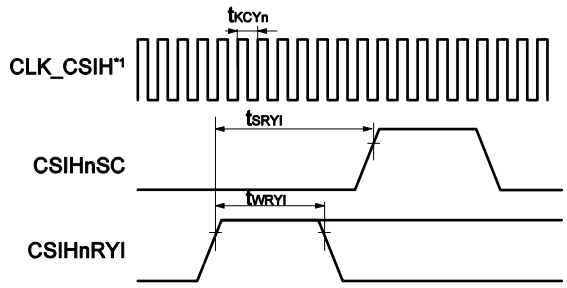

**Note 1.** CLK_CSIH is internal clock.

**Figure 35.20 CSIH Master Mode Timing – CSS[7:0] hold (CSIHnSIT = 1)**




**Note 1.** CLK_CSIH is internal clock.

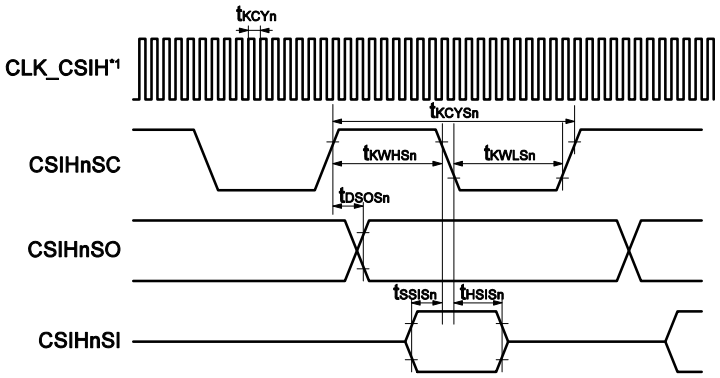
**Figure 35.21 CSIH Master Mode Timing – CSS[7:0] RYI (CSIHnCKP0 = 0)**




**Note 1.** CLK_CSIH is internal clock.

**Figure 35.22 CSIH Master Mode Timing – CSS[7:0] RYI (CSIHnCKP0 = 1)**

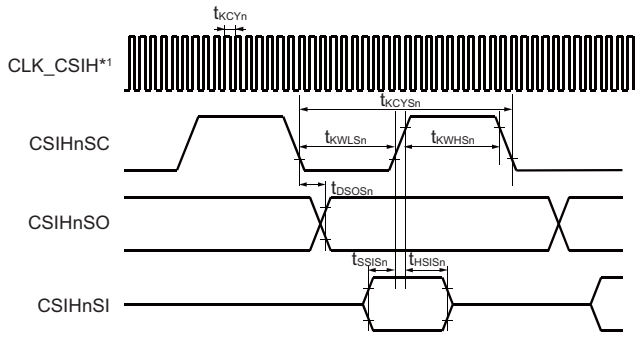
### (2) CSIH Waveform (Slave Mode)




**Note 1.** CLK_CSIH is internal clock.

**Note:** CSIHnCKR = 0, CSIHnCKP0 = 0, CSIHnDAP0 = 1 or CSIHnCKR = 0, CSIHnCKP0 = 1, CSIHnDAP0 = 0 or CSIHnCKR = 1, CSIHnCKP0 = 0, CSIHnDAP0 = 0

**Figure 35.23 CSIH Slave Mode Timing – General Definition**




**Note 1.** CLK_CSIH is internal clock.

**Note:** CSIHnCKR = 0, CSIHnCKP0 = 0, CSIHnDAP0 = 0 or CSIHnCKR = 0, CSIHnCKP0 = 1, CSIHnDAP0 = 1 or CSIHnCKR = 1, CSIHnCKP0 = 0, CSIHnDAP0 = 1

**Figure 35.24 CSIH Slave Mode Timing – General Definition**

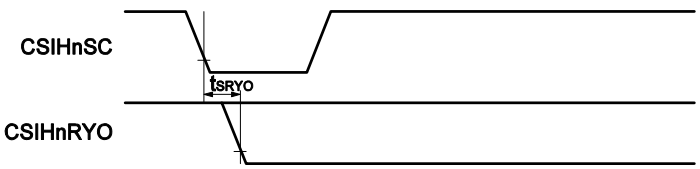



**Figure 35.25 CSIH Slave Mode Timing – RYO (CSIHnCKP0 = 0, CSIHDAP0 = 0)**

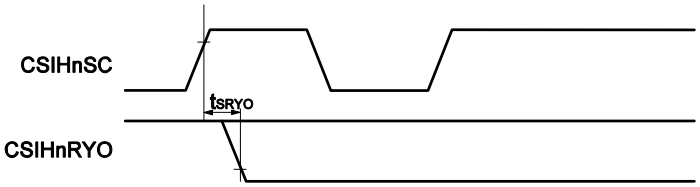



**Figure 35.26 CSIH Slave Mode Timing – RYO (CSIHnCKP = 0, CSIHDAP0 = 1)**

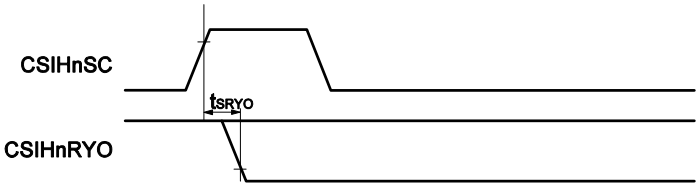



**Figure 35.27 CSIH Slave Mode Timing – RYO (CSIHnCKP0 = 1, CSIHDAP0 = 0)**

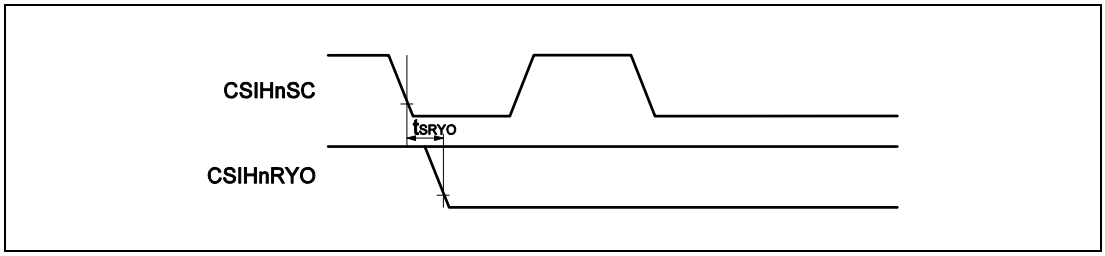
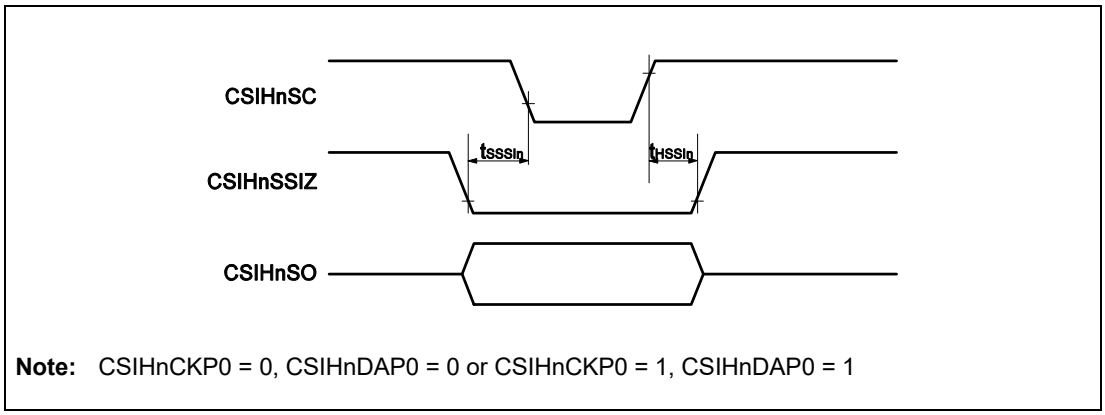
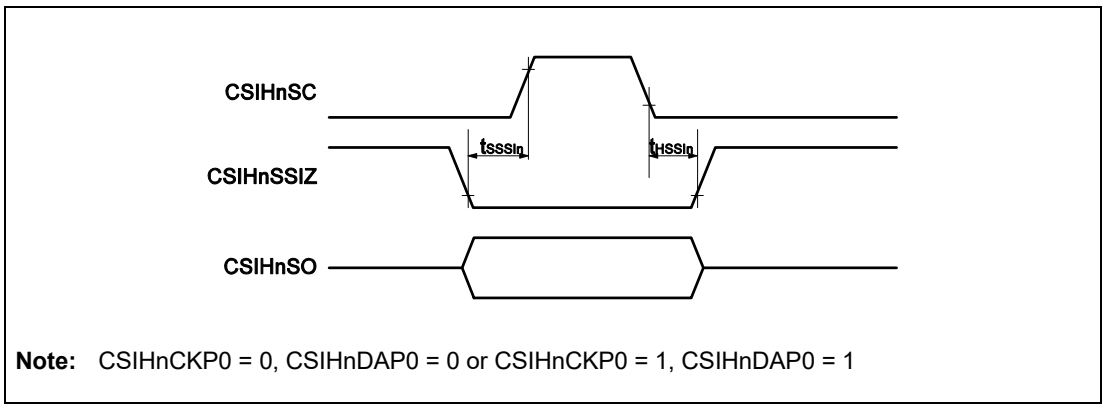



Figure 35.28 CSIH Slave Mode Timing – RYO (CSIHnCKP0 = 1, CSIHDAP0 = 1)




**Note:** CSIHnCKP0 = 0, CSIHnDAP0 = 0 or CSIHnCKP0 = 1, CSIHnDAP0 = 1

Figure 35.29 CSIH Slave Mode Timing – SSI




**Note:** CSIHnCKP0 = 0, CSIHnDAP0 = 0 or CSIHnCKP0 = 1, CSIHnDAP0 = 1

Figure 35.30 CSIH Slave Mode Timing – SSI

## 35.6.10 FlexRay Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- The following conditions are different from the general ones given above:
  - Capacitive load (Cload) connected to output pins: Up to 25pF
    It is necessary to select 50 Ω output buffer.
  - EnVCC = 3.0 to 3.6V.

**Table 35.30 FlexRay Timing*1**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Transfer rate | RFLX | FLXnRXDA, FLXnRXDB, 10Mbps | | | 10 | Mbps |

Note 1. Base of this specification is “FlexRay Electrical Physical Layer Specification V3.0.1, Oct-2010”

**NOTE**

“n” means unit number of Flexray. n = 0 or 1

## 35.6.11 RLIN3 Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**

**Table 35.31 RLIN3 Timing**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| RLIN3 transfer rate | RRLlin | LIN mode | | | 20 | kbps |
| | RLurt | UART mode | | | 6.6 | Mbps |

## 35.6.12 Ethernet Timing

### 35.6.12.1 MII Characteristics

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- Capacitive load (Cload) connected to output pins: Up to 15pF
  It is necessary to select 50 Ω output buffer.

**Table 35.32 Ethernet Timing – 100 Mbit/s MII Characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| ETHmTXCLK width | $t_{MTC}$ | | 40 – 100 ppm | 40 | 40 + 100 ppm | ns |
| ETHmTXCLK high width | $t_{MTCH}$ | *1 | 14 | 20 | 26 | ns |
| ETHmTXCLK low width | $t_{MTCL}$ | *1 | 14 | 20 | 26 | ns |
| ETHmTXD [3:0] delay time | $t_{MTXD}$ | | 0 | | 25 | ns |
| ETHmTXEN delay time | $t_{MTXE}$ | | 0 | | 25 | ns |
| ETHmRXCLK width | $t_{MRC}$ | | 40 – 100 ppm | 40 | 40 + 100 ppm | ns |
| ETHmRXCLK high width | $t_{MRCH}$ | *1 | 14 | 20 | 26 | ns |
| ETHmRXCLK low width | $t_{MRCL}$ | *1 | 14 | 20 | 26 | ns |
| ETHmRXD [3:0] setup time | $t_{MRXDS}$ | | 10 | | | ns |
| ETHmRXD [3:0] hold time | $t_{MRXDH}$ | | 10 | | | ns |
| ETHmRXDV, ETHmRXER setup time | $t_{MRDES}$ | | 10 | | | ns |
| ETHmRXDV, ETHmRXER hold time | $t_{MRDEH}$ | | 10 | | | ns |

Note 1. The duty cycle of ETHmTXCLK and ETHmRXCLK shall be between 35 to 60% inclusive. (IEEE802.3)

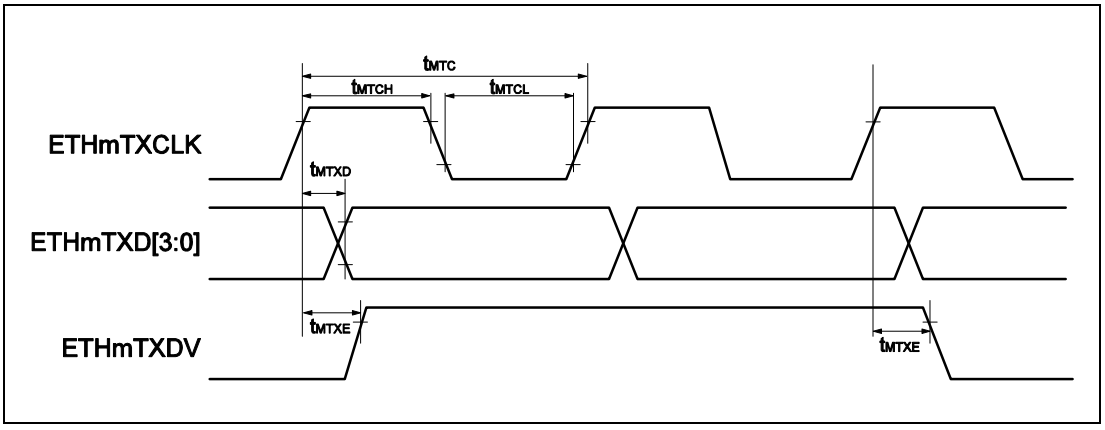



**Figure 35.31 Ether net Timing – MII Transmitter**

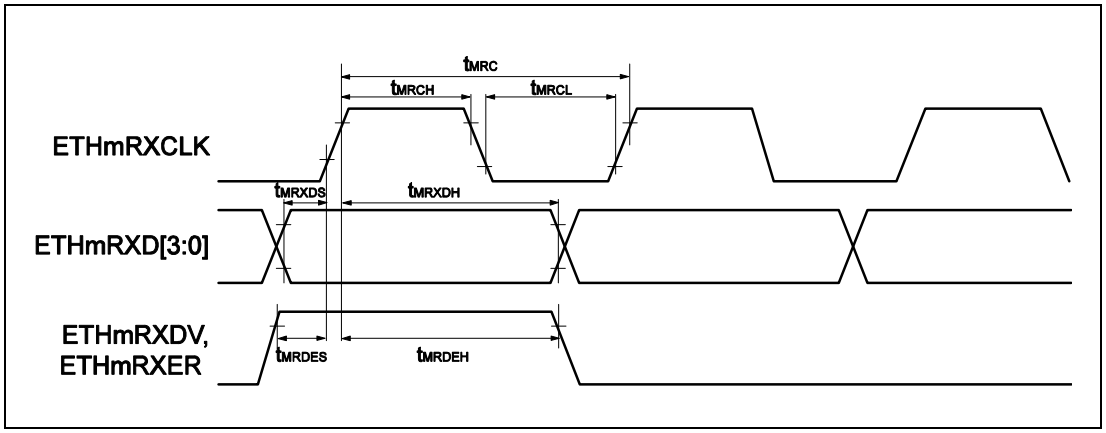



Figure 35.32 Ethernet Timing – MII Receiver

### 35.6.12.2 RMII Characteristics

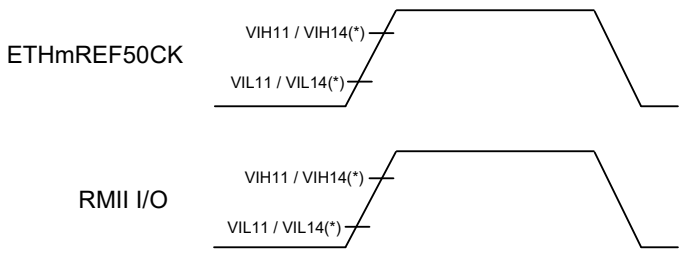
**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- Capacitive load (Cload) connected to output pins: Up to 15pF
  It is necessary to select 20 Ω output buffer.

Table 35.33 Ethernet Timing – RMII Characteristics

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| ETHmREF50CK width | $t_{RMC}$ | | 20 – 50 ppm | 20 | 20 + 50 ppm | ns |
| ETHmREF50CK high width | $t_{RMCH}$ | *1 | 7 | | 13 | ns |
| ETHmREF50CK low width | $t_{RMCL}$ | *1 | 7 | | 13 | ns |
| ETHmRXD [1:0], ETHmCRS and ETHmRXER setup time | $t_{RMS}$ | | 4 | | | ns |
| ETHmRXD [1:0], ETHmCRS and ETHmRXER hold time | $t_{RMH}$ | | 2 | | | ns |
| ETHmTXD [1:0], ETHmTXEN output delay time | $t_{RMD}$ | Cload < 15 pF | 2 | | 16 | ns |

Note 1. The dutry cycle of ETHmREF50CK shall be between 35% to 65% inclusive. (RMII™ specification)




**Note 1.** VIH and VIL specification depend on PISn/JPIS0 register setting. Refer to **Section 2 Pin Functions**

**Figure 35.33 Ethernet Timing – RMII AC Measurements (Valid I/O Levels)**

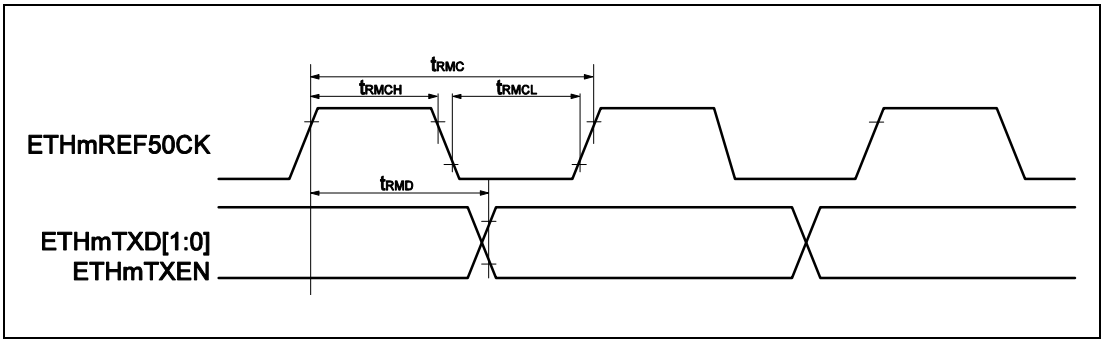



**Figure 35.34 Ethernet Timing – RMII Transmitter**

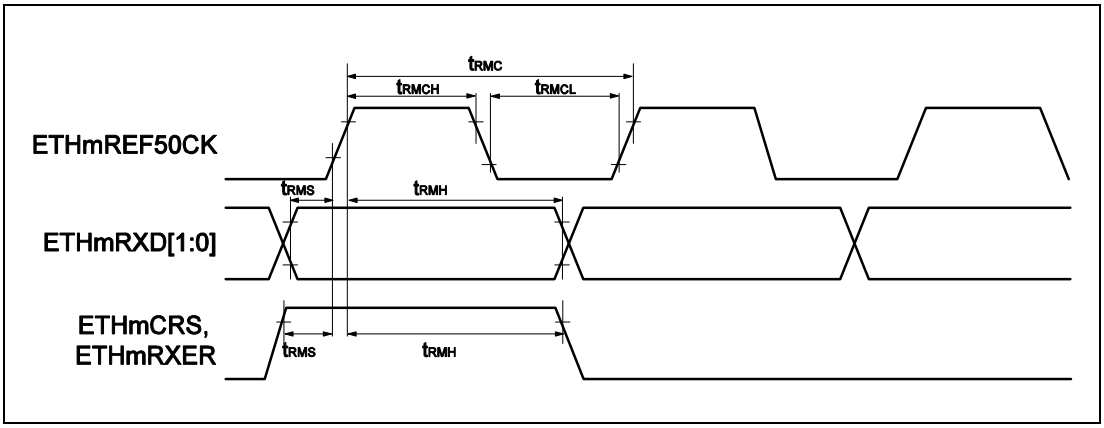



**Figure 35.35 Ethernet Timing – RMII Receiver**

## 35.6.13 GTM Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- Below terminals have no “100 Ω” output buffer.
  P3_9 (GTMAT1O0), P3_10 (GTMAT0O1), P3_14 (GTMAT0O1),
  P8_9 (GTMAT2O0), P8_10 (GTMAT20N), P8_11 (GTMAT2O1)
  So please select “50 Ω” or “200 Ω” output buffer about these terminals.

**Table 35.34 GTM Timing**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| GTM input high-level width | $t_{WGTH}$ | | 2 × $T_{samp}$ *1 | | | ns |
| GTM input low-level width | $t_{WGTL}$ | | 2 × $T_{samp}$ *1 | | | ns |
| GTM output cycle time | $t_{CYGT}$ | | 4 × $T_{samp}$ *1 | | | ns |

Note 1. $T_{samp} = 1 / f_{CLK_HSB}$

## 35.6.14 MCAN Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- Below terminals have no “100 Ω” output buffer.
  P8_10 (MTTCAN0SOC), P8_11 (MTTCAN0TMP)
  So please select “50 Ω” or “200 Ω” output buffer about these terminals.

**Table 35.35 MCAN Timing**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Transfer rate*1 | RMCN1 | CAN-FD arbitration and CAN 2.0B | | | 1 | Mbps |
| | RMCN2*2 | CAN-FD data phase | | | 8 | Mbps |
| Internal delay time (input plus output) | tDMCIN | tDIN + tDOUT (Cload = 15pF, with 50 Ω output buffer) | | | 30 | ns |

Note 1. Protocol layer clock needs to be configured appropriately.
Note 2. It is necessary to select 50 Ω output buffer.

## 35.6.15 External Memory Controller (MEMC0) Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- The following conditions are different from the general ones given above:
  - Please use internal pull-up for external memory interface data signal MEMC0D[7:0]I / MEMC0D[7:0]O
  - Below terminals have no "100 Ω" output buffer.
    P8_9 (MEMC0D1O), P8_10 (MEMC0D2O), P8_11 (MEMC0D3O)
    So please select "50 Ω" output buffer about these terminals.

### 35.6.15.1 MEMC0 — Read Access Timing (50 Ω output buffer / 30pF).

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- Capacitive load (Cload) connected to output pins: Up to 30pF
  It is necessary to select 50 Ω output buffer.

**Table 35.36 MEMC0 – Read Access Timing (50 Ω output buffer / 30pF)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Bus operational period | T | Max. 20 MHz | 50 | | | ns |
| MEMC0D[7:0] data input setup time (from _MEMC0RDZ↓) | $t_{SRDID}$ | | | | T(*1) – 29 | ns |
| MEMC0RDZ low level width | $t_{WRDL}$ | | T(*1) – 19 | | | ns |
| MEMC0A[8:0] address, MEMC0CS[3:0]Z →MEMC0RDZ↓ delay time | $t_{DARD}$ | | T – 19 | | | ns |
| MEMC0RDZ↑→from MEMC0A[8:0] address, MEMC0CS[3:0]Z delay time | $t_{DRDA}$ | | –15 | | | ns |
| MEMC0D[7:0] data input hold time (from MEMC0RDZ↑) | $t_{HRDID}$ | | 0 | | | ns |
| MEMC0RDZ↑→MEMC0D[7:0] data output delay time | $t_{DRDOD}$ | | T(*2) – 9 | | | ns |

Note 1. Adjustment by WD is possible. $W_D$: wait cycle determined by DWC0 register.
Note 2. Adjustment by IWR is possible. $I_{WR}$: number of idle state after read cycle.

### 35.6.15.2 MEMC0 — Read Access Timing (100 Ω output buffer / 30pF).

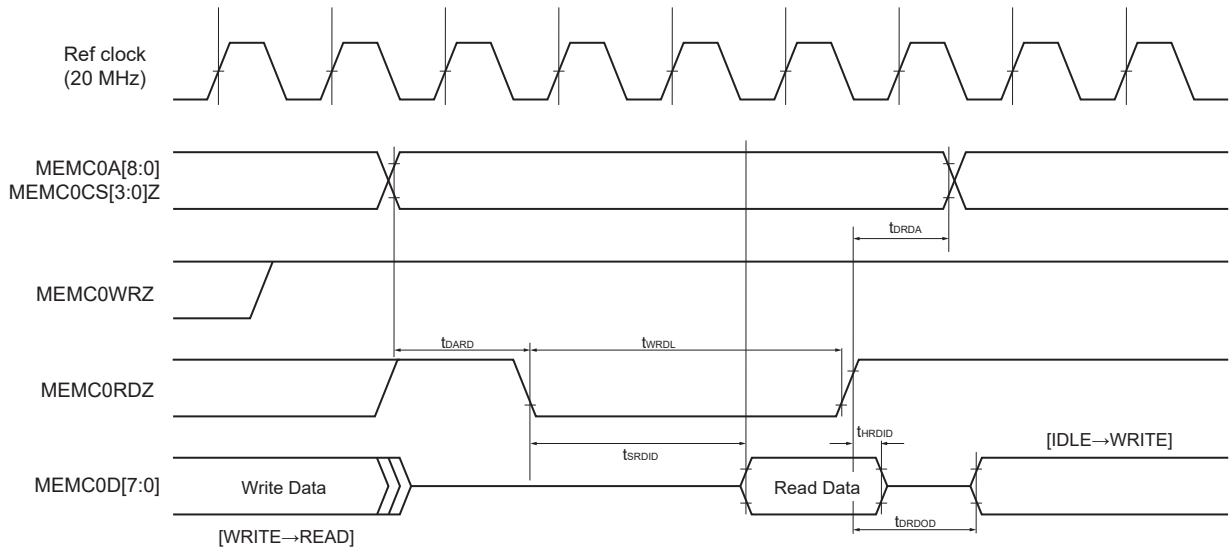
**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- Capacitive load (Cload) connected to output pins: Up to 30pF
  Output buffer can be selected 50/100 Ω output buffer.

**Table 35.37 MEMEC0 – Read Access Timing (100 Ω output buffer / 30pF)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Bus operational period | T | Max. 20 MHz | 50 | | | ns |
| MEMC0D[7:0] data input setup time (from _MEMC0RDZ↓) | $t_{SRDID}$ | | | | T(*1) – 38 | ns |
| MEMC0RDZ low level width | $t_{WRDL}$ | | T(*1) – 19 | | | ns |
| MEMC0A[8:0] address, MEMC0CS[3:0]Z →MEMC0RDZ↓ delay time | $t_{DARD}$ | | T – 25.5 | | | ns |
| MEMC0RDZ↑→from MEMC0A[8:0] address, MEMC0CS[3:0]Z delay time | $t_{DRDA}$ | | –15 | | | ns |
| MEMC0D[7:0] data input hold time (from MEMC0RDZ↑) | $t_{HRDID}$ | | 0 | | | ns |
| MEMC0RDZ↑→MEMC0D[7:0] data output delay time | $t_{DRDOD}$ | | T(*2) – 9 | | | ns |

Note 1. Adjustment by WD is possible. $W_D$: wait cycle determined by DWC0 register.
Note 2. Adjustment by IWR is possible. $I_{WR}$: number of idle state after read cycle.




**Note:** The dashed lines indicates high impedance.

**Figure 35.36 MEMC0 – READ Access Timing**

### 35.6.15.3 MEMC0 — Write Access Timing (50 Ω output buffer / 30pF)

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- Capacitive load (Cload) connected to output pins: Up to 30pF
  It is necessary to select 50 Ω output buffer.

**Table 35.38 MEMC0 – Write Access Timing (50 Ω output buffer / 30pF)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Bus operational period | T | Max. 20 MHz | 50 | | | ns |
| MEMC0A[8:0] address, MEMC0CS[3:0]Z delay time (from _MEMC0WRZ↓) | $t_{DAWR}$ | | T – 19 | | | ns |
| MEMC0WRZ↑→MEMC0A[8:0] address, MEMC0CS[3:0]Z delay time | $t_{DWRA}$ | | T(*2) – 25 | | | ns |
| MEMC0WRZ low level width | $t_{WWRL}$ | | T(*1) – 19 | | | ns |
| MEMC0D[7:0] data output setup time (from MEMC0WRZ↑) | $t_{SODWR1}$ | WRITE2 | 2T(*1) – 19 | | | ns |
| MEMC0D[7:0] data output setup time (from MEMC0WRZ↑) | $t_{SODWR2}$ | WRITE1 | 2T(*1) – 19 | | | ns |
| MEMC0D[7:0] data output hold time (from MEMC0WRZ↑) | $t_{HWROD1}$ | WRITE2 | T(*2) – 9 | | | ns |
| MEMC0D[7:0] data output hold time (from MEMC0WRZ↑) | $t_{HWROD2}$ | WRITE1 | T(*2) – 9 | | | ns |

Note 1. Adjustment by WD is possible. $W_D$: Wait cycle determined by DWC0 register.
Note 2. Adjustment by WDH is possible. $W_{DH}$: wait cycle determined by DHC register.

### 35.6.15.4 MEMC0 — Write Access Timing (100 Ω output buffer / 30pF)

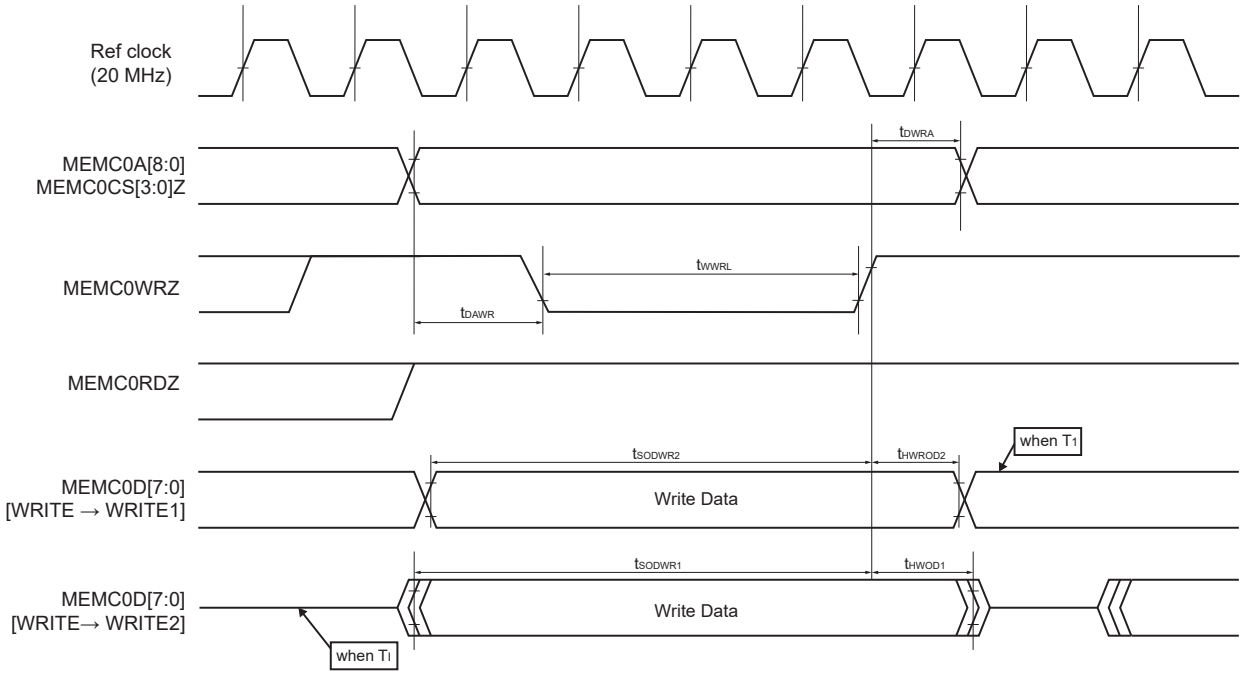
**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- Capacitive load (Cload) connected to output pins: Up to 30pF
  Output buffer can be selected 50/100 Ω output buffer.

**Table 35.39 MEMC0 – Write Access Timing (100 Ω output buffer / 30pF)**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Bus operational period | T | Max. 20 MHz | 50 | | | ns |
| MEMC0A[8:0] address, MEMC0CS[3:0]Z delay time (from _MEMC0WRZ↓) | $t_{DAWR}$ | | T – 25.5 | | | ns |
| MEMC0WRZ↑→MEMC0A[8:0] address, MEMC0CS[3:0]Z delay time | $t_{DWRA}$ | | T(*2) – 25 | | | ns |
| MEMC0WRZ low level width | $t_{WWRL}$ | | T(*1) – 19 | | | ns |
| MEMC0D[7:0] data output setup time (from MEMC0WRZ↑) | $t_{SODWR1}$ | WRITE2 | 2T(*1) – 25.5 | | | ns |
| MEMC0D[7:0] data output setup time (from MEMC0WRZ↑) | $t_{SODWR2}$ | WRITE1 | 2T(*1) – 25.5 | | | ns |
| MEMC0D[7:0] data output hold time (from MEMC0WRZ↑) | $t_{HWROD1}$ | WRITE2 | T(*2) – 9 | | | ns |
| MEMC0D[7:0] data output hold time (from MEMC0WRZ↑) | $t_{HWROD2}$ | WRITE1 | T(*2) – 9 | | | ns |

Note 1. Adjustment by WD is possible. $W_D$: Wait cycle determined by DWC0 register.
Note 2. Adjustment by WDH is possible. $W_{DH}$: wait cycle determined by DHC register.




**Note 1.** In case of WRITE → WRITE operation
If TI cycle is not inserted, MEMC0 operates as [WRITE → WRITE1]
If TI cycle is inserted, MEMC0 operates as [WRITE → WRITE2]

**Note 2.** The dashed lines indicates high impedance.

**Figure 35.37 MEMC0 – Write Access Timing**

## 35.6.16 Emergency Shut-Off (ESO) Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**

**Table 35.40 ESO Timing**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Delay time ESOnZ ↓ to GTMATnOx HiZ | $t_{DESO}$ | | | | 50 | ns |

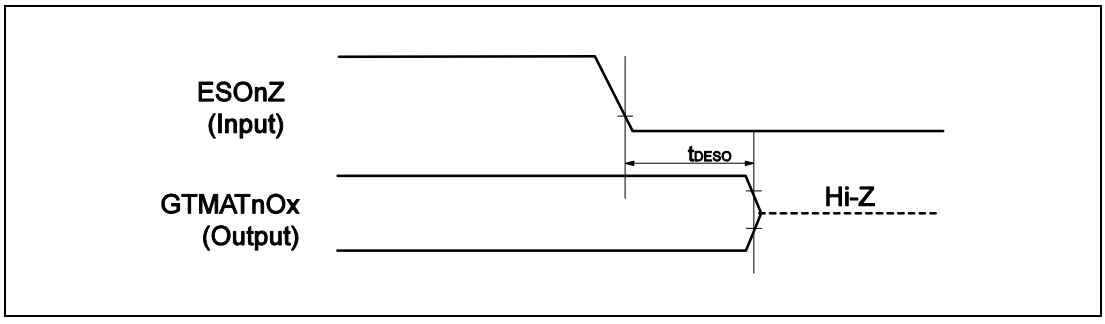



**Figure 35.38 ESO Timing**

### 35.6.17 TRSTZ Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**

**Table 35.41 TRSTZ Timing**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| TRSTZ input low level width | $t_{WTRL}$ | | 600 | | | ns |
| TRSTZ release timing delay | $t_{DTRT2R}$ | | 30 | | | ms |

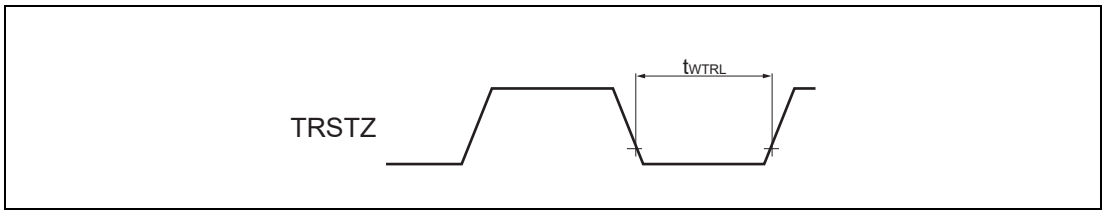



**Figure 35.39 TRSTZ low level timing**

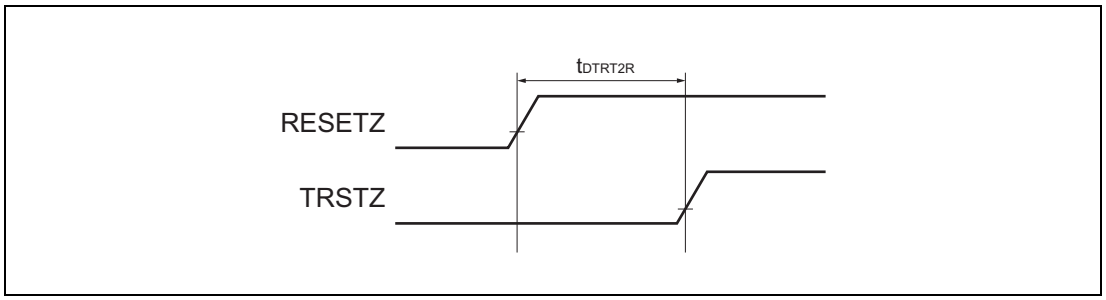



**Figure 35.40 TRSTZ release timing**

## 35.6.18 Nexus Interface Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions** except output buffer. Output buffer is selected 50Ω.

**Table 35.42 Nexus Interface Timing**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| TCK Cycle width | $t_{DCKW}$ | | 50 | | 666 | ns |
| TCK high level width | $t_{CKWH}$ | | 21 | | | ns |
| TCK low level width | $t_{CKWL}$ | | 21 | | | ns |
| TDI setup time (vs. TCK ↑) | $t_{SDI}$ | | 40 | | | ns |
| TDI hold time (vs. TCK ↑) | $t_{HDI}$ | | 3 | | | ns |
| TMS setup time (vs. TCK ↑) | $t_{SMS}$ | | 40 | | | ns |
| TNS hold time (vs. TCK ↑) | $t_{HMS}$ | | 3 | | | ns |
| TDO delay time (vs. TCK ↓) | $t_{DDO}$ | | 0 | | 20 | ns |
| RDYZ delay time (vs. TCK ↓) | $t_{RDYZ}$ | | 0 | | 20 | ns |

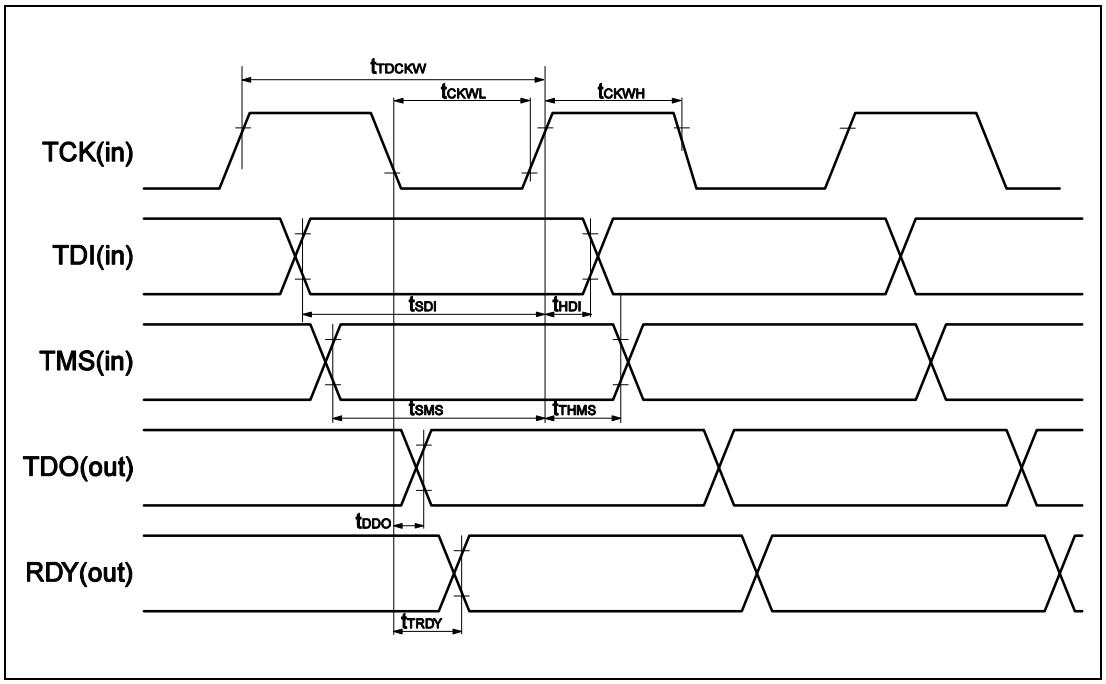



**Figure 35.41 NEXUS Interface Timing**

## 35.6.19 LPD (4pin) Interface Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions** except output buffer. Output buffer is selected 50Ω.

**Table 35.43 LPD (4pin) Interface Timing**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| LPDCLK cycle time / LPDCLKOUT cycle time | $t_{LPDCLKCY}$ | | 83.3 (max. 12MHz) | | 666 | ns |
| LPDCLK high level width / LPDCLK low level width | $t_{LPDCKW}$ | | 0.5 × $t_{LPDCLKCY}$ – 10 | | | ns |
| LPDCLKOUT high level width / LPDCLKOUT low level width | $t_{LPDCKOW}$ | | $t_{LPDCKW}$ – 10 | | | ns |
| LPDI setup time (vs. LPDCLK ↑) | $t_{LPDIS}$ | | 41 | | | ns |
| LPDI hold time (vs. LPDCLK ↑) | $t_{LPDIH}$ | | 3 | | | ns |
| LPDCLK to LPDCLKOUT delay time | $t_{LPDCKOD}$ | | | | 44 | ns |
| LPDO delay time (vs. LPDCLKOUT ↑) | $t_{LPDOD}$ | | 0 | | 15 | ns |

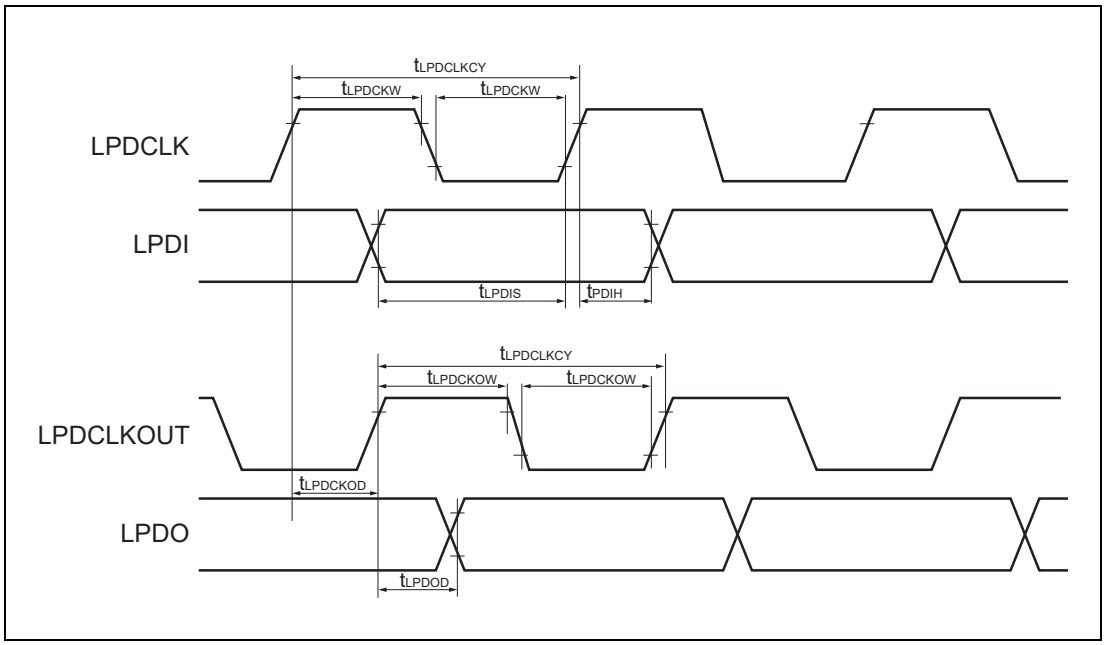



**Figure 35.42 LPD (4pin) Interface Timing**

## 35.6.20 AUDR Interface Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**
- Capacitive load (Cload) connected to output pins: 10pF

**Table 35.44 AUDR Interface Timing**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| AUDCK cycle time | $t_{AUCKcyc}$ | | 20 | | | ns |
| AUDCK high level width | $t_{AUCKH}$ | | 0.4 | | | $t_{AUCKcyc}$ |
| AUDCK low level width | $t_{AUCKL}$ | | 0.4 | | | $t_{AUCKcyc}$ |
| AUDRSTZ setup time | $t_{AURSTS}$ | | 12 | | | ns |
| AUDRSTZ pulse width | $t_{AURSTW}$ | | 5 | | | $t_{AUCKcyc}$ |
| AUDSYNCZ setup time | $t_{AUSYS}$ | | 10 | | | ns |
| AUDSYNCZ hold time | $t_{AUSYH}$ | | 5 | | | ns |
| AUDATA input setup time | $t_{AUDTS}$ | | 8 | | | ns |
| AUDATA input hold time | $t_{AUDTH}$ | | 5 | | | ns |
| AUDATA output delay | $t_{AUDTD}$ | | | | $t_{AUCKcyc}$ – 3.4 | ns |

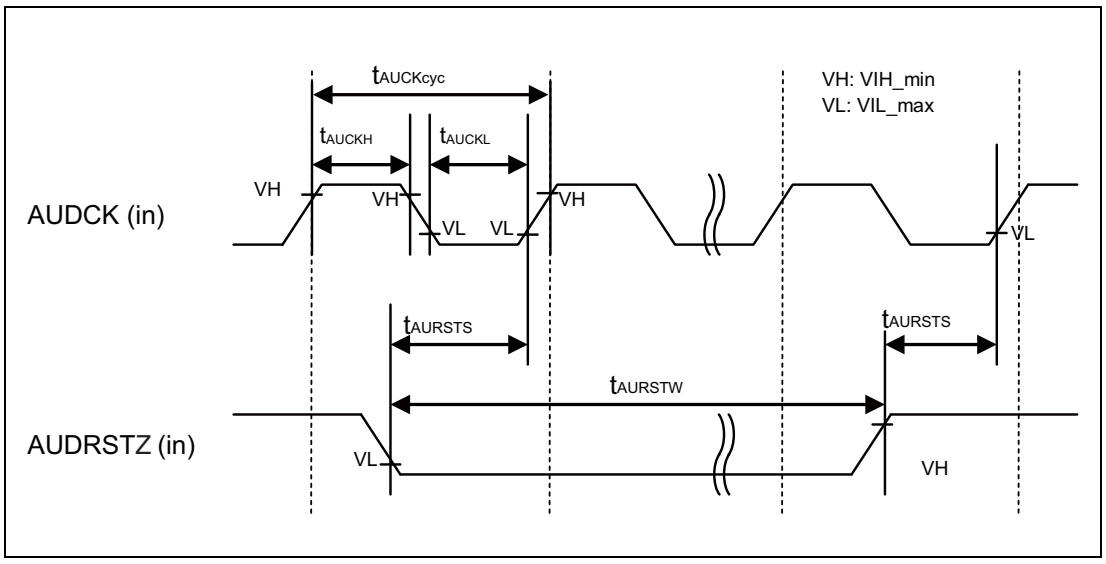



**Figure 35.43 AUDRSTZ Timing**

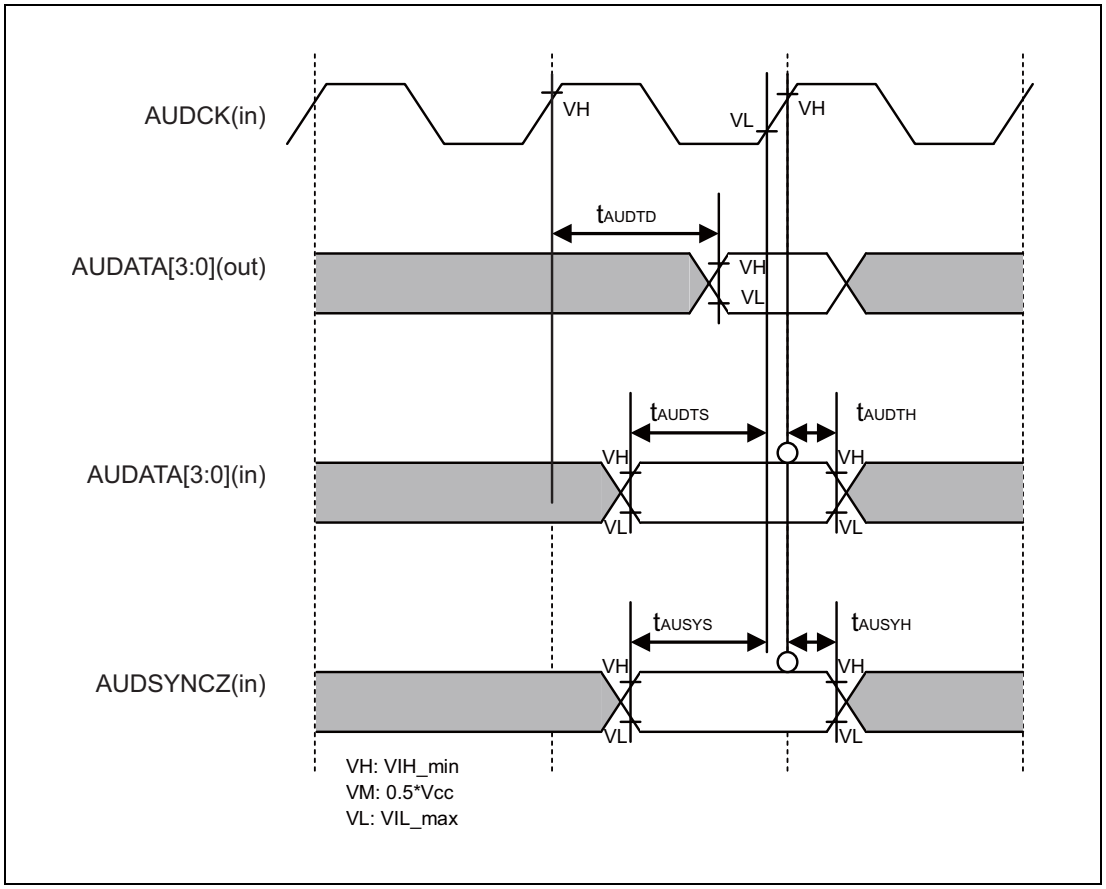



**Figure 35.44 AUDR AC Timing**

## 35.6.21 Aurora Interface Timing

**Conditions:**

- See **Section 35.6.1.1, General Conditions**

### 35.6.21.1 Aurora Interface — 1.25 Gbps baud rate

**Table 35.45 Aurora Interface Operating Condition – 1.25Gbps baud rate**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Differential Voltage | VDIFF | *1 | 800 | | 1600 | mV |
| Rise / Fall Time | TRF | *2 | 60 | | | ps |
| Deterministic jitter | JD | | | | 0.17 | UI |
| Total jitter | JT | | | | 0.35 | UI |
| Output skew | SO | *3 | | | 25 | ps |
| Multiple output skew | SMO | *4 | | | 1000 | ps |
| Unit interval | UI | ± 100 ppm | 800 | | 800 | ps |
| Differential output resistance | ZOD | | 70 | | 130 | Ω |

Note 1. Peak to peak differential.
Note 2. At driver output.
Note 3. Skew at transmitter output between the differential pair.
Note 4. Skew at transmitter output between lanes of a multi-lane channel.

### 35.6.21.2 Aurora Interface — 2.5 Gbps baud rate

**Table 35.46 Aurora Interface Operating Condition – 2.5Gbps baud rate**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Differential Voltage | VDIFF | *1 | 800 | | 1600 | mV |
| Rise / Fall Time | TRF | *2 | 40 | | | ps |
| Deterministic jitter | JD | | | | 0.17 | UI |
| Total jitter | JT | | | | 0.35 | UI |
| Output skew | SO | *3 | | | 20 | ps |
| Multiple output skew | SMO | *4 | | | 1000 | ps |
| Unit interval | UI | ± 100 ppm | 400 | | 400 | ps |
| Differential output resistance | ZOD | | 70 | | 130 | Ω |

Note 1. Peak to peak differential.
Note 2. At driver output.
Note 3. Skew at transmitter output between the differential pair.
Note 4. Skew at transmitter output between lanes of a multi-lane channel.

### 35.6.21.3 Aurora Interface — 3.125 Gbps baud rate

Table 35.47 Aurora Interface Operating Condition – 3.125Gps baud rate

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Differential Voltage | VDIFF | *1 | 800 | | 1600 | mV |
| Rise / Fall Time | TRF | *2 | 30 | | | ps |
| Deterministic jitter | JD | | | | 0.17 | UI |
| Total jitter | JT | | | | 0.35 | UI |
| Output skew | SO | *3 | | | 15 | ps |
| Multiple output skew | SMO | *4 | | | 1000 | ps |
| Unit interval | UI | ± 100 ppm | 320 | | 320 | ps |
| Differential output resistance | ZOD | | 70 | | 130 | Ω |

Note 1. Peak to peak differential.
Note 2. At driver output.
Note 3. Skew at transmitter output between the differential pair.
Note 4. Skew at transmitter output between lanes of a multi-lane channel.

### 35.6.21.4 Aurora Interface — Transmitter clock Timing

Table 35.48 Aurora Interface Transmitter clock timing

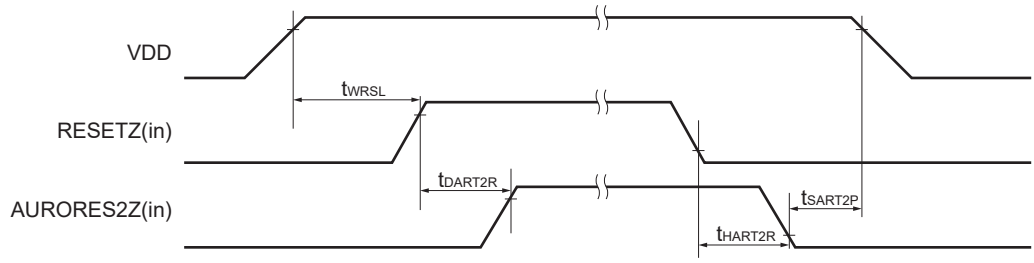
| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Reference clock frequency | CLKP | 1.25 Gbps baud rate | | | 62.5 | MHz |
| | | 2.5 Gbps baud rate | | | 125 | MHz |
| | | 3.125 Gbps baud rate | | | 156.25 | MHz |
| Reference clock rise time | | | | | 400 | ps |
| Reference clock fall time | | | | | 400 | ps |
| Reference clock duty cycle | | | 45 | | 55 | % |
| Reference clock total jitter | | *1 | | | 40 | ps |
| Stability | | | 50 | | | ppm |

Note 1. Peak to peak.

## 35.6.22 Debug Resource (Aurora, ERAM) Specific Reset Timing

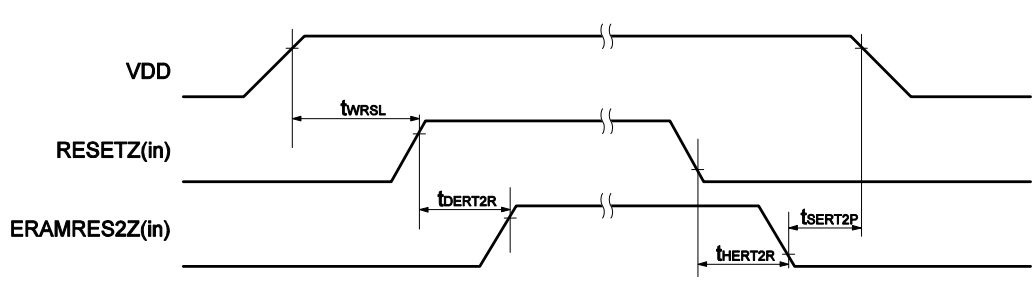
Table 35.49 Debug Resource (Aurora, ERAM) Specific Reset Timing

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| AURORES2Z release timing delay | $t_{DART2R}$ | | 0 | | 400 | ns |
| AURORES2Z inactive hold time | $t_{HART2R}$ | | 2.4 | | | µs |
| AURORES2Z setup time | $t_{SART2P}$ | | 0 | | | ns |
| ERAMRES2Z release timing delay | $t_{DERT2R}$ | | 0 | | 400 | ns |
| ERAMRES2Z inactive hold time | $t_{HERT2R}$ | | 2.4 | | | µs |
| ERAMRES2Z assert setup time | $t_{SERT2P}$ | | 0 | | | ns |
| AURORES1Z release timing delay | $t_{DART2AVD}$ | | 2.4 | | | µs |
| AUROREST1Z inactive hold time | $t_{SART2AVD}$ | | 2.4 | | | µs |
| ERAMRESPDZ release timing delay | $t_{DERT2EVD}$ | | 2.4 | | | µs |
| ERAMRESPDZ inactive hold time | $t_{SERT2EVD}$ | | 2.4 | | | µs |




**Note:** About the parameter $t_{WRSL}$, please refer to “RESETZ low pulse width” in **Section 35.6.3**

Figure 35.45 AURORES2Z AC Timing




**Note:** About the parameter $t_{WRSL}$, please refer to “RESETZ low pulse width” in **Section 35.6.3**

Figure 35.46 ERAMRES2Z AC Timing

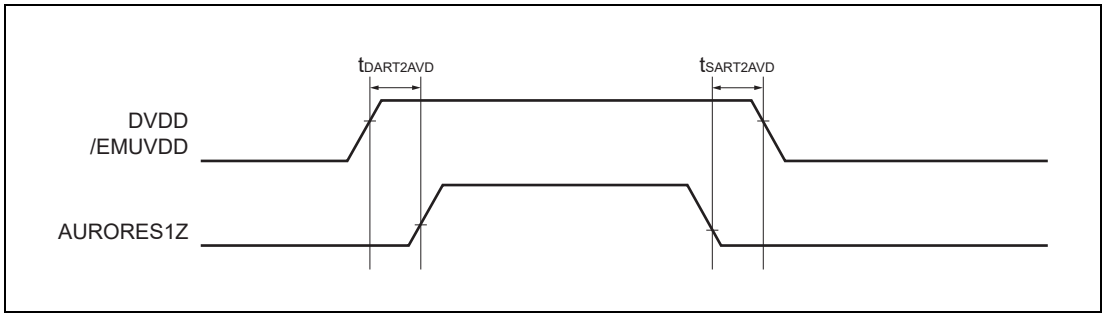



**Figure 35.47 AURORES1Z AC Timing**

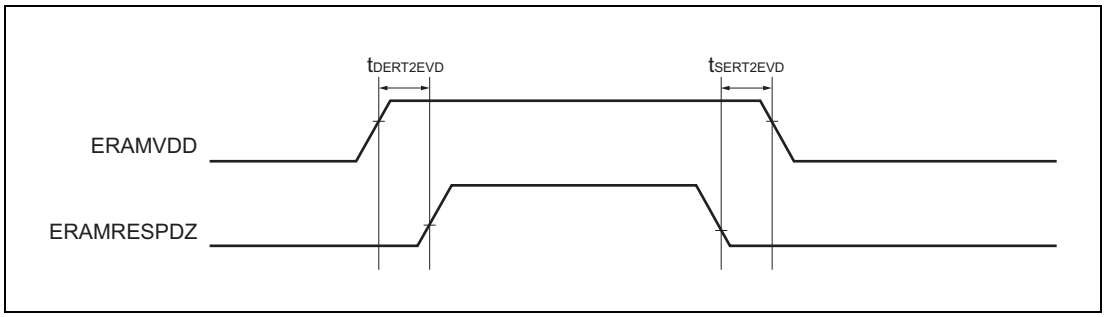



**Figure 35.48 ERAMRESPDZ AC Timing**

## 35.6.23 Debug Event Interface Timing

Table 35.50 Debug Event Interface Timing

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| EVTIZ input high level width | $t_{WEVIH}$ | | $t_{CPUCLK}$ *2 | | | ns |
| EVTIZ input low level width | $t_{WEVIL}$ | | $t_{CPUCLK}$ *2 | | | ns |
| EVTOZ output high level width | $t_{WEVOH}$ | *1 | 16 × $t_{CPUCLK6}$ – 10 | 60 | | ns |
| EVTOZ output low level width | $t_{WEVOL}$ | *1 | 16 × $t_{CPUCLK6}$ – 10 | 60 | | ns |
| MSYNZ input high level width | $t_{WMSNH}$ | | 40 | | | ns |
| MSYNZ input low level width | $t_{WMSNL}$ | | 40 | | | ns |

Note 1. Base clock frequency is CPU operating clock frequency ($f_{CPUCLK}$). Typical value corresponds to the case $f_{CPUCLK}$ = 240 MHz.

Note 2. $t_{CPUCLK}$ indicates a period of the lowest CPU clock in the device.

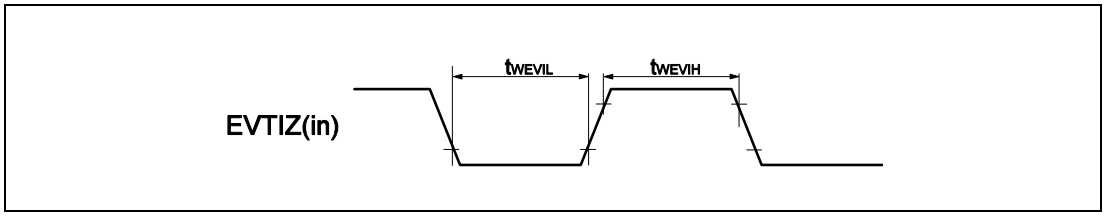



Figure 35.49 EVITZ AC Timing

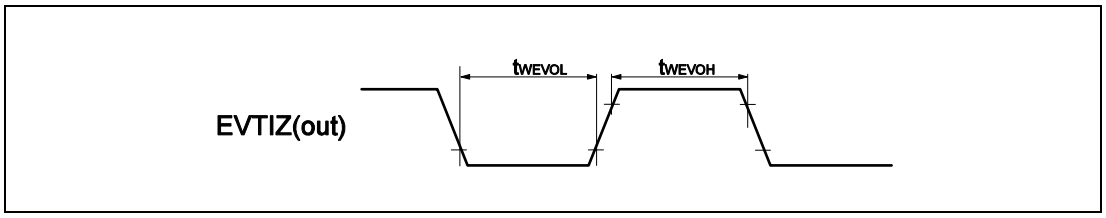



Figure 35.50 EVOTZ AC Timing

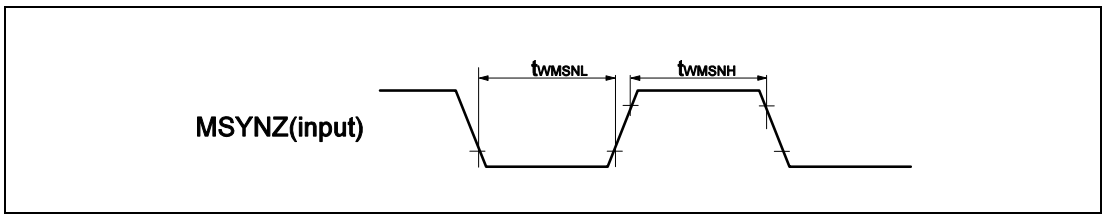



Figure 35.51 MSYNZ AC Timing

## 35.6.24 Flash Programming Characteristics

**Conditions:**

- See **Section 35.6.1.1, General Conditions** except output buffer. Output buffer is selected 50Ω.

**Table 35.51 Flash FLSCI programming characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| FLSCI3 transfer rate | | 1-wired UART mode ($f_{CLK_LSB}$ = 40MHz) | | | 1 | Mbps |
| | | 2-wired UART mode ($f_{CLK_LSB}$ = 40MHz) | | | 2 | Mbps |
| FLSCI3SCKI cycle time | $t_{KCYSF}$ | 3-wired Clock Sync mode ($f_{CLK_LSB}$ = 40MHz) | 200*1 | | | ns |
| FLSCI3SCKI high level width | $t_{KWHSF}$ | 3-wired Clock Sync mode | $t_{KCYSF}$ / 2 – 15 | | | ns |
| FLSCI3SCKI low level width | $t_{KWLSF}$ | 3-wired Clock Sync mode | $t_{KCYSF}$ / 2 – 15 | | | ns |
| FLSCI3RXD setup time (vs. FLSCI3SCKI) | $t_{SSISF}$ | 3-wired Clock Sync mode | 2 × $t_{Pcyc}$*2 | | | ns |
| FLSCI3RXD hold time (vs. FLSCI3SCKI) | $t_{HSISF}$ | 3-wired Clock Sync mode | 2 × $t_{Pcyc}$*2 | | | ns |
| FLSCI3TXD output delay (vs. FLSCI3SCKI) | $t_{DSOSF}$ | 3-wired Clock Sync mode | 2 × $t_{Pcyc}$*2 | | 3 × $t_{Pcyc}$*2 + 36 | ns |

Note 1. Input the external clock data is more than 6 clocks of CLK_LSB.
Note 2. $t_{Pcyc}$ is a period of CLK_LSB

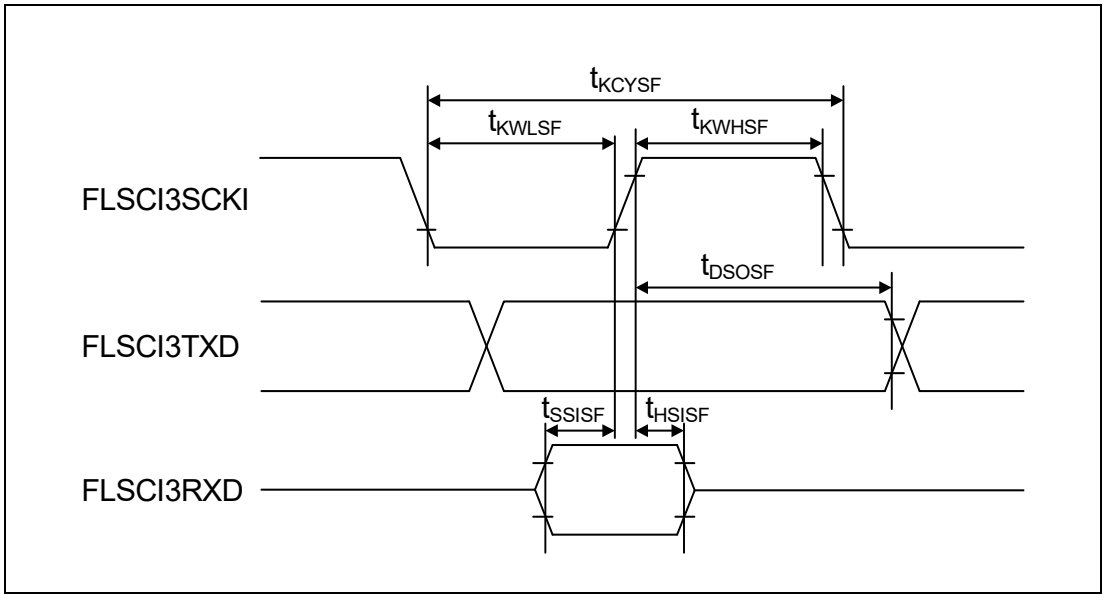



**Figure 35.52 FLSCI Timing**

**Table 35.52 Serial Programmer Setup Timing**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Delay time VCC ↑ & SYSVCC ↑ & IOVCC ↑ to RESETZ ↑ | $t_{DPOR}$ | | $t_{MOST}$ *1 | | | ms |
| FLMD0, 1 setup time (vs RESETZ ↑) | $t_{SMDR}$ *3 | | 1 | | | ms |
| FLMD0, 1 hold time (vs RESETZ ↑) | $t_{HMDR}$ *3 | | 1 | | | ms |
| FLMD0 pulse input start time | $t_{RP}$ | | 900 | | | µs |
| FLMD0 pulse input end time | $t_{RPE}$ | | | | 11.5 | ms |
| FLMD0 high level width | $t_{PWH}$ | | $t_{WFDH}$ *2 | | | ns |
| FLMD0 low level width | $t_{PWL}$ | | $t_{WFDL}$ *2 | | | ns |
| FLMD0 rise time | $t_R$ | | | | 1 | µs |
| FLMD0 fall time | $t_F$ | | | | 1 | µs |

Note 1. For $t_{MOST}$ refer to **Section 35.3.2, Main Oscillator Characteristics**.
Note 2. For $t_{PWH}$/$t_{PWL}$ refer to **Section 35.6.5, Mode Timing**
Note 3. Refer to **Section 35.6.2, Power Up/Down Timing**

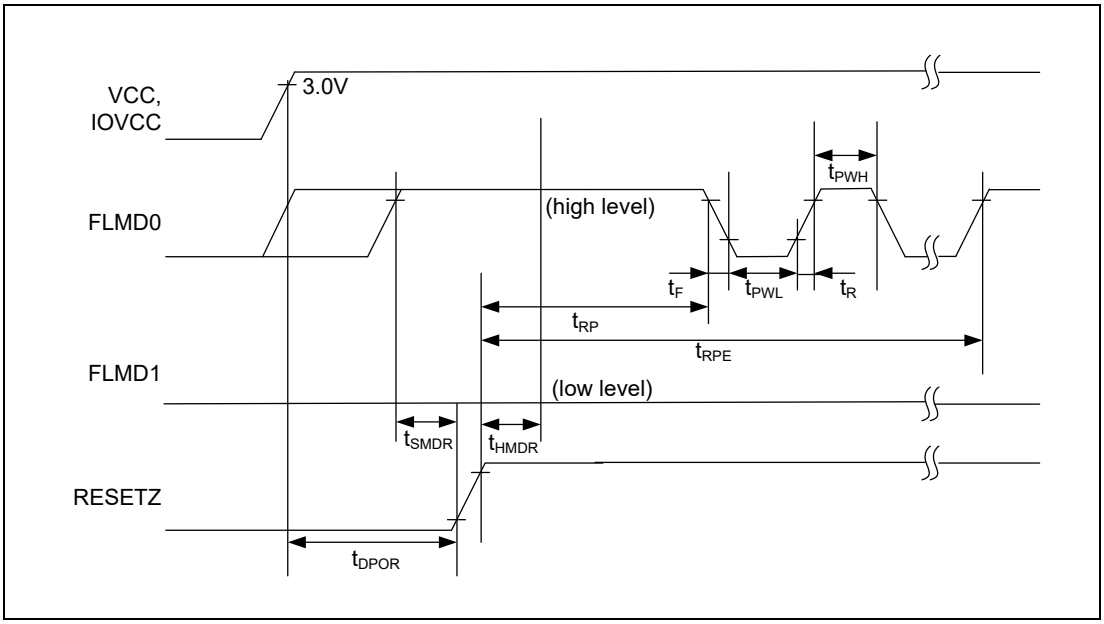



**Figure 35.53 Serial Programmer Setup Timing**

# 35.7 Analog Functions Characteristics

## 35.7.1 POC Characteristics

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.53 POC Characteristics*1**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Detection threshold voltage | $V_{POC}$ | Rise | 2.70 | 2.85 | 3.00 | V |
| | | Fall | 2.70 | 2.80 | 2.90 | V |
| POC delay time | $t_{DPOC}$ | | | | 2 | ms |
| POC minimum pulse width | $t_{WPOC}$ | | 0.2 | | | ms |

Note 1. POC monitors SYSVCC supply voltage.

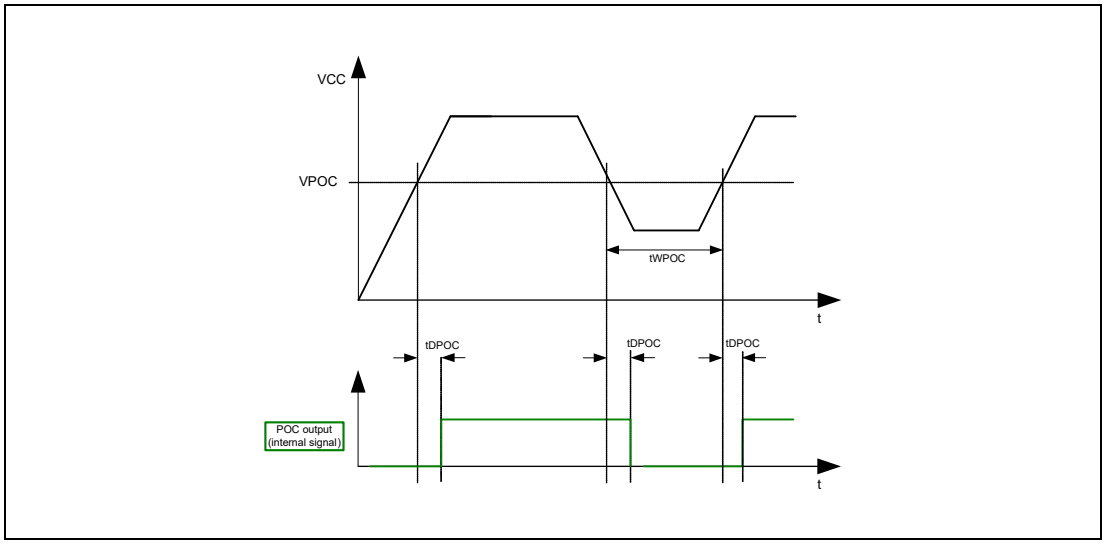



**Figure 35.54 POC Characteristics**

### 35.7.2 Core Voltage Monitor (CVM) Characteristics

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.54 CVM Characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| CVM high detection level | $V_{CVMH}$ | | 1.35 | 1.39 | 1.43 | V |
| CVM low detection level | $V_{CVML}$ | | 1.10 | 1.15 | 1.20 | V |
| CVM delay time | $t_{DCVM}$ | | 0.2 | | 10 | µs |
| CVM filter time | $t_{FCVM}$ | | 2 | 4 | 6 | µs |

## 35.7.3 A/D Converter Characteristics

**Conditions:**

- Temperature range: Tjmin to Tjmax.
- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

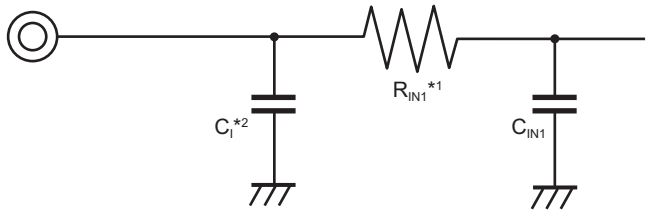
**Table 35.55 ADC Characteristics**

| Item | Symbol | Condition | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Resolution | | | | | 12 | | Bit |
| Analog supply voltages | AnVCC | | | 3.0 | | 3.6 | V |
| | | No guarantee of accuracy required, guarantee of operation is mandatory. | | $V_{POC}$ | | 3.0 | V |
| Reference voltages | AnVREFH | | | 3.0 | | AnVCC | V |
| | | No guarantee of accuracy required, guarantee of operation is mandatory. | | $V_{POC}$ | | 3.0 | V |
| Analog input voltage | $V_{IAN}$ | | | AnVSS | | AnVREFH | V |
| Operation frequency | $f_{ADCLK}$ | | | 8 | | 40 | MHz |
| A0VCC current | $I_{A0VCC}$ | | | | | 4 | mA |
| A1VCC current | $I_{A1VCC}$ | | | | | 4 | mA |
| A0VREFH current | $I_{A0VREFH}$ | | | | | 0.5 | mA |
| A1VREFH current | $I_{A1VREFH}$ | | | | | 0.5 | mA |
| Power up time w/o T&H | $t_{PU}$ | | | | | 1.0 | µs |
| Conversion time (standard) | $t_{CONV1}$ | $t_{CONV}$ includes $t_{SAMP}$ | | 1.0 | | | µs |
| Sample time | $t_{SAMP}$ | | | $t_{CONV1}$ × (18/40) | | | ns |
| Total overall error | TOE | Include quantization error | | –6.0 | | +6.0 | LSB |
| Integral non-linearity error | INL | | | –3.0 | | +3.0 | LSB |
| Differential non-linearity error | DNL | No missing code | | –1.0 | | +2.0 | LSB |
| Offset error | OSE | | | –5.5 | | +5.5 | LSB |
| Full-scale error | FSE | | | –5.5 | | +5.5 | LSB |
| Self-diagnosis accuracy | TOEdiag | A/D Conversion Circuit Diagnosis on single ADC*2 | Include quantization error | –8.0 | | +8.0 | LSB |
| | | Pin Level Diagnosis from I/O*1 | | –40.0 | | +40.0 | LSB |
| Equivalent input resistance w/o T&H | RIN1*3 | PIN – RRanp: 96.7%<br>PIN – 12 bit SAR – ADC: 3.3% | | | | 3.1 | kΩ |
| Equivalent input sampling capacitance w/o T&H | CIN1 | | | | | 5.8 | pF |
| TDE switch resistance | RTDE*3 | ADCFnTDCR.TDE = 0 | | 100 | | 600 | Ω |

Note 1. The accuracy of the conversion result for the applied diagnostic voltage could be out of specification in case injected current is injected to the pin of this channel

Note 2. For details to operate “A/D Conversion Circuit Diagnostic Function” refer to CAUTION on **Section 27.4.7.2, A/D Conversion Circuit Diagnostic Function**

Note 3. RIN1 includes RTDE. For detail of TDE switch, see **Figure 27.11, diagnosis circuits**.




Note 1. RIN1 includes RTDE. For detail of TDE switch, see **Figure 27.11, diagnosis circuits**.
Note 2. For CI data, see **Section 35.3.6, IO Capacitances**.

**Figure 35.55 Equivalent Circuit**

### 35.7.4 Temperature Sensor Characteristics

**Conditions:**

- Supply voltage range: Refer to **Section 35.3.1, Supply Voltage Characteristics**.
- System control supply voltage range: SYSVCC = 3.0V to 3.6V.
- EnVSS = AnVSS = OSCVSS = VSS.
- Reference ground potential: VSS = 0 V.

**Table 35.56 Temperature Sensor (TS) Characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Temperature range | TTS1 | *2 | Tjmin | | Tjmax*1 | °C |
| Temperature accuracy*3 | ACCTS1 | | –3 | | +3 | °C |
| Temperature update period | $t_{TSUP}$ | | | | 10 | ms |
| Stand-by return time | $t_{TSSB1}$ | | | | 200 | µs |
| Operation frequency | $f_{OTSCLK}$*4 | | 36 | | 40 | MHz |

Note 1. Value of Tjmax is device dependent, see par. **Table 35.5 Operational Condition**.

Note 2. The operation of Temperature Sensor itself and ICUMC reaction for thermal attack between Tjmax and Tj=165degC is guaranteed, but by design only and will not be tested.

Note 3. For an absolute temperature within the given accuracy an offset has to be considered. The offset is given by trimming data in the register OTS0COEFFRA to OTS0COEFFRC.

Note 4. For $f_{OTSCLK}$, refer to **Section 11.1.3, Clock Supply**.

## 35.7.5 Flash Characteristics

### (1) Code Flash

The code flash memory is shipped in the erased state. If the code flash memory is read where it has not been written after erasure (no write condition), an ECC error is generated, resulting in the occurrence of an exception.

**Table 35.57 Basic Characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Operation frequency | $f_{CLK_LSB}$ | | 2 | | 40 | MHz |
| Operation voltage | VDD | | 1.20 | | 1.35 | V |
| | VCC | | 3.0 | | 3.6 | V |
| Number of rewrites*1 | CWRT | Data retention of 20 years*2 | 1000 | | | times |
| Programming temperature (Tj) | TPRG | | Tjmin*3 | | Tjmax*3 | °C |

Note 1. The number of rewrites is the number of erasures for each block. When the number of rewrites is “n” (n = 1000), the device can be erased “n” times for each block. For example, when a block of 32 KB is erased after 256 bytes of writing have been performed for different addresses 128 times, the number of rewrites is counted as 1. However, multiple writing to the same address is not possible with 1 erasure (overwriting prohibited).

Note 2. This is the case when the average Ta is 85°C. This is the period starting on completion of a successful erasure of the code flash memory.

Note 3. Refer to **Table 35.5 Operational Condition**

**Table 35.58 Programming Characteristics**

| Item | Symbol | Condition | Block size | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Programming time*1 | | $f_{CLK_LSB}$ ≥ 20 MHz<br>CWRT < 100 times | 256 B | | 2 | 6 | ms |
| | | $f_{CLK_LSB}$ ≥ 20 MHz<br>CWRT ≥ 100 times | 256 B | | 2.4 | 7.2 | ms |
| | | $f_{CLK_LSB}$ ≥ 20 MHz | 8 KB | | 16 | 96 | ms |
| | | $f_{CLK_LSB}$ ≥ 20 MHz | 32 KB | | 64 | 384 | ms |
| | | $f_{CLK_LSB}$ ≥ 20 MHz<br>CWRT < 10 times<br>Tj = 10 – 80 °C | 512 KB | | 0.8 | 3.6 | s |
| | | $f_{CLK_LSB}$ ≥ 20 MHz<br>CWRT ≥ 100 times | 512 KB | | 1.03 | 6.15 | s |
| Erasure time | | $f_{CLK_LSB}$ ≥ 20 MHz | 8 KB | | 47 | 235 | ms |
| | | $f_{CLK_LSB}$ ≥ 20 MHz | 32 KB | | 169 | 576 | ms |
| | | $f_{CLK_LSB}$ ≥ 20 MHz<br>CWRT < 10 times<br>Tj = 10 – 80 °C | 512 KB | | 2.3 | 4.5 | s |
| | | $f_{CLK_LSB}$ ≥ 20 MHz<br>CWRT ≥ 100 times | 512 KB | | 2.71 | 8.4 | s |

Note 1. Timings are based on availability of write buffer according to programmed block size of either 256 B, 8 KB or 32 KB.

**Table 35.59 Program/Erase suspend latency Characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Program Suspend Latency*1 | | $f_{CLK_LSB}$ ≥ 20 MHz | | | 120 | µs |
| Erase Suspend Latency*1*2 | | Suspension-Priority (1st suspension for a pulse application) $f_{CLK_LSB}$ ≥ 20 MHz | | | 120 | µs |
| | | Suspension-Priority (2nd suspension for a pulse application) $f_{CLK_LSB}$ ≥ 20 MHz | | | 1.7 | ms |
| | | Erasure-Priority $f_{CLK_LSB}$ ≥ 20 MHz | | | 1.7 | ms |
| Program Resume Latency*1 | | $f_{CLK_LSB}$ ≥ 20 MHz | | | 50 | µs |
| Erase Resume Latency*1 | | Suspension-Priority (after 1st suspension) $f_{CLK_LSB}$ ≥ 20 MHz | | | 1.7 | ms |
| | | Suspension-Priority (after 2nd suspension) $f_{CLK_LSB}$ ≥ 20 MHz | | | 80 | µs |
| | | Erasure-Priority $f_{CLK_LSB}$ ≥ 20 MHz | | | 80 | µs |

Note 1. The target value is the hardware suspend time. A possible software overhead is not considered.

Note 2. FACI command (Code Flash Erase) with erase counter update cannot be suspended until erase counter update has finished.

### (2) Data Flash

The data flash memory is shipped in the erased state. If the data flash memory is read where it has not been written after erasure (no write condition), an ECC error is generated, resulting in the occurrence of an exception

**Table 35.60 Basic Characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Operation frequency | $f_{CLK_LSB}$ | | 2 | | 40 | MHz |
| Operation voltage | VDD | | 1.20 | | 1.35 | V |
| | VCC | | 3.0 | | 3.6 | V |
| Number of rewrites*1 | CWRT | Data retention 20 years*2 | 125 k | | | times |
| | | Data retention 3 years*2 | 250 k | | | times |
| Programming temperature (Tj) | TPRG | | Tjmin*3 | | Tjmax*3 | °C |

Note 1. The number of rewrites is the number of erasures for each block. When the number of rewrites is “n” (n = 125000), the device can be erased “n” times for each block. For example, when a block of 64 bytes is erased after 4 bytes of writing have been performed for different addresses 168 times, the number of rewrites is counted as 1. However, multiple writing to the same address is not possible with 1 erasure (overwriting prohibited).

Note 2. This is the case when the average Ta is 85°C. This is the period starting on completion of a successful erasure of the data flash memory.

Note 3. Refer to **Table 35.5 Operational Condition**.

**Table 35.61 Programming Characteristics**

| Item | Symbol | Condition | Block size | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Programming time | | $f_{CLK_LSB}$ ≥ 20 MHz | 4 B | | 0.16 | 1.7 | ms |
| Erasure time | | $f_{CLK_LSB}$ ≥ 20 MHz<br>CWRT < 10 times<br>Tj = 10 – 80 °C | 8 KB | | 216 | 640 | ms |
| | | $f_{CLK_LSB}$ ≥ 20 MHz | 64 B | | 1.7 | 10 | ms |
| Blank check time | | $f_{CLK_LSB}$ ≥ 20 MHz | 4 B | | | 30 | µs |

**Table 35.62 Program/Erase suspend latency Characteristics**

| Item | Symbol | Condition | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Program Suspend Latency*1 | | $f_{CLK_LSB}$ ≥ 20 MHz | | | 120 | µs |
| Erase Suspend Latency*1 | | Suspension-Priority (1st suspension for a pulse application) $f_{CLK_LSB}$ ≥ 20 MHz | | | 120 | µs |
| | | Suspension-Priority (2nd suspension for a pulse application) $f_{CLK_LSB}$ ≥ 20 MHz | | | 300 | µs |
| | | Erasure-Priority $f_{CLK_LSB}$ ≥ 20 MHz | | | 300 | µs |
| Program Resume Latency*1 | | $f_{CLK_LSB}$ ≥ 20 MHz | | | 50 | µs |
| Erase Resume Latency*1 | | Suspension-Priority (after 1st suspension) $f_{CLK_LSB}$ ≥ 20 MHz | | | 300 | µs |
| | | Suspension-Priority (after 2nd suspension) $f_{CLK_LSB}$ ≥ 20 MHz | | | 70 | µs |
| | | Erasure-Priority $f_{CLK_LSB}$ ≥ 20 MHz | | | 70 | µs |

Note 1. The target value is the hardware suspend time. A possible software overhead is not considered.

## 35.8 Thermal Characteristics

The chip junction temperature TJ, max must never be exceeded. Please refer to the device and package dependent specification of TJ, max below.

It is the responsibility of the user to make sure the maximum junction temperature is not exceeded.

The actual TJ depends on the power dissipation and the thermal characteristics of the application. Please refer to chapter **Section 35.5, Supply Current Characteristics**, how to calculate the power dissipation. Power dissipation can be influenced by choosing several parameters. The thermal characteristic of the application environment need to be considered throughout the design process. The following sub-chapters describe metrics to consider this.

### 35.8.1 Junction-to-Board Resistance (Ψjb)

The simplest method to determine the actual chip temperature is to use the single resistance metric of Ψjb.

The following equation may be used:

$$T_j = T_b + (P_{TOTn} \times Y_{jb})$$

with:

- $T_J$: chip junction temperature in [°C]
- $T_b$: board temperature (according to JEDEC standard JESD51-2A) in [°C]
- $P_{tot}$: total power consumption (refer to **Section 35.5, Supply Current Characteristics**) in [W]
- Ψjb: thermal resistance between junction and board in [°C/W]

This simple metric considers the test board properties in a natural convection environment. The thermal resistance is derived from a defined test fixture (JEDEC) or simulation of such test fixture using a 3D simulation with a detailed model.

**Table 35.63 Thermal Characteristics - Junction to Board Resistance (Ψjb)**

| Device | Symbol*1 | Package | Condition*2*3 | MAX Value | Unit |
|---|---|---|---|---|---|
| P1M-C | Ψjb32 | BGA292 | JEDEC, JESD51-9, TJ, max = 150°C | 13.2*5 | °C/W |
| | Ψjb34 | BGA156 | JEDEC, JESD51-9, TJ, max = 150°C | 13.3 | °C/W |
| | Ψjb33 | LQFP144 | JEDEC, JESD51-7, TJ, max = 160°C | 26.7*4 | °C/W |
| P1H-C (4MB) | Ψjb41 | BGA292 | JEDEC, JESD51-9, TJ, max = 150 °C | 10.0*5 | °C/W |
| | Ψjb42 | BGA156 | JEDEC, JESD51-9, TJ, max = 150 °C | 10.2 | °C/W |
| P1H-C (8MB) | Ψjb51 | BGA292 | JEDEC, JESD51-9, TJ, max = 150 °C | 8.2*5 | °C/W |
| P1H-CE | Ψjb71 | BGA404 | JEDEC, JESD51-9, TJ, max = 150 °C | 5.1*5 | °C/W |

Note 1. In case of LQFP, Ψjb is a thermal resistance between junction and board surface at tip of LQFP pin. In case of BGA, Ψjb is a thermal resistance between junction and board surface beneath center of package

Note 2. Thermal resistance is based on a solder saturation (solder wetting) >= 50 %.

Note 3. Total power of Digital IO Pins and Analog Input Pins is calculated by assuming 60mW for P1M-C and P1H-C.
Note 4. Rth Tolerance: +2degC/W, L/F: standard L/F
Note 5. Data includes max Rth Tolerance

# Section 36 Package

## 36.1 Package Line Up

| RH850/P1x-C Product | P1M-C 144 | P1M-C 156, P1H-C 156 | P1M-C 292, P1H-C 292 | P1H-CE |
|---|---|---|---|---|
| Renesas code | PLQP0144LB-A | PLBG0156JB-A | PRBG0292GC-A | PRBG0404GA-A |
| JEITA code | P-LFQFP144-16x16-0.40 | P-FBGA156-10x10-0.65 | P-FBGA292-17x17-0.80 | P-FBGA404-19x19-0.80 |
| Name | LQFP | FBGA | FBGA | FBGA |
| Terminal count | 144 | 156 | 292 | 404 |
| Terminal pitch (mm) | 0.4 | 0.65 | 0.8 | 0.8 |
| Dimensions (mm) | 16x16 | 10x10 | 17x17 | 19x19 |
| Mass (g)[TYP] | 0.9 | 0.37 | 0.95 | 1.1 |
| Mounting height (mm)[MAX] | 1.7 | 1.7 | 1.9 | 1.9 |
| Terminal material - Base | Cu alloy | Sn-Ag-Cu | Sn-Ag-Cu | Sn-Ag-Cu |
| Environment | Lead free | Lead free | Lead free | Lead free |

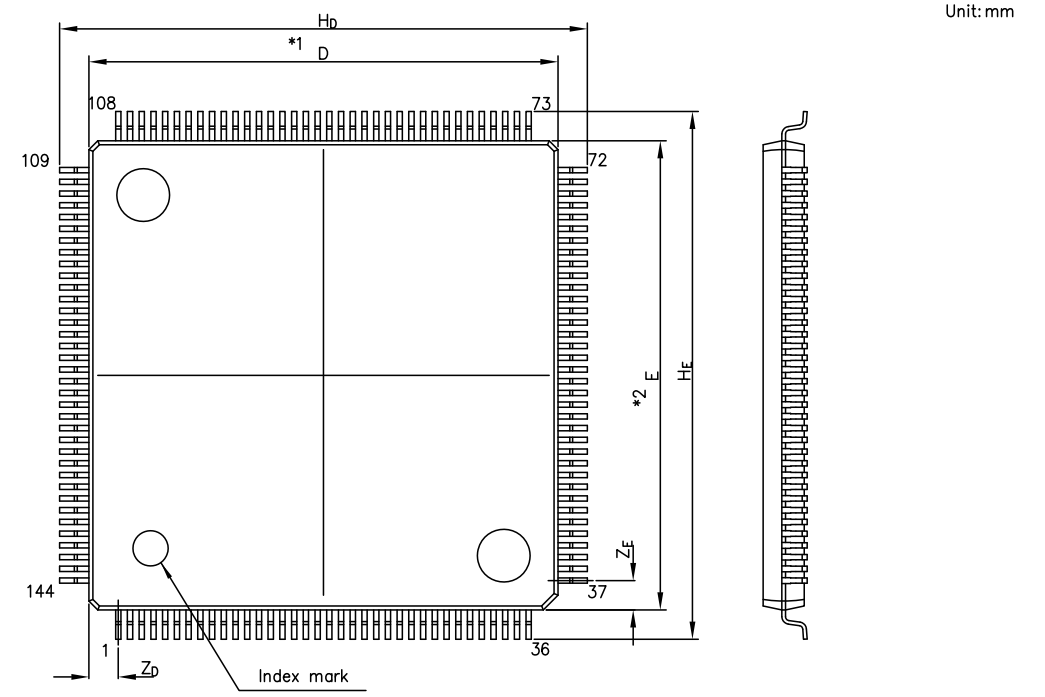
## 36.2 Package Outline

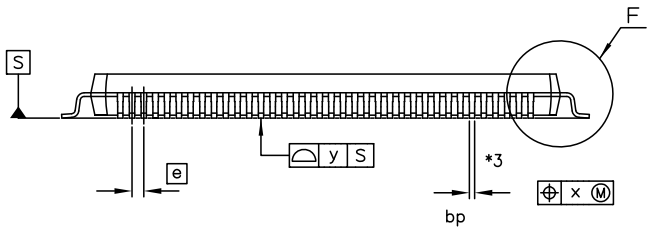
### 36.2.1 LQFP Package Drawing

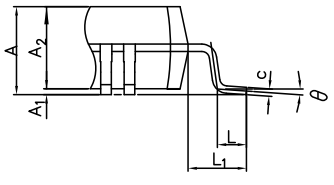
- **Figure 36.1** shows LQFP outline.

| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-LFQFP144-16x16-0.40 | PLQP0144LB-A | —— | 0.9g |

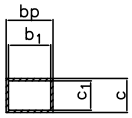
Unit: mm




Detail F




Terminal cross section

NOTE)
1. DIMENSIONS "*1" AND "*2" DO NOT INCLUDE MOLD FLASH.
2. DIMENSION "*3" DOES NOT INCLUDE TRIM OFFSET.

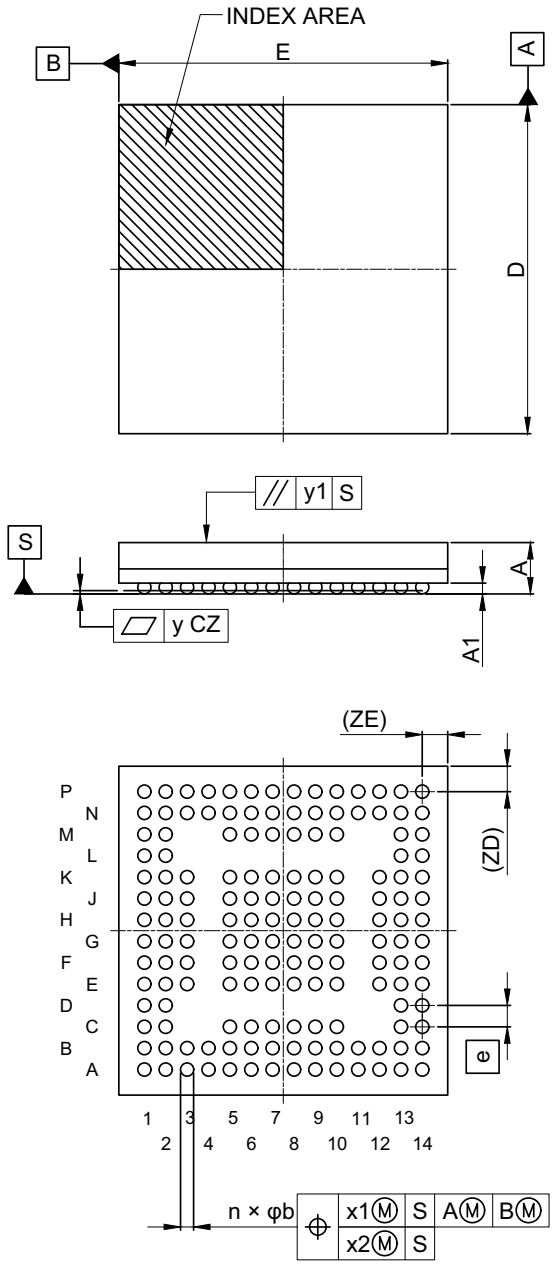
| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | 15.9 | 16.0 | 16.1 |
| E | 15.9 | 16.0 | 16.1 |
| A2 | —— | 1.4 | —— |
| HD | 17.8 | 18.0 | 18.2 |
| HE | 17.8 | 18.0 | 18.2 |
| A | —— | —— | 1.7 |
| A1 | 0.05 | 0.10 | 0.15 |
| bp | 0.13 | 0.18 | 0.23 |
| b1 | —— | 0.16 | —— |
| c | 0.09 | 0.145 | 0.20 |
| c1 | —— | 0.125 | —— |
| θ | 0° | —— | 8° |
| e | —— | 0.4 | —— |
| x | —— | —— | 0.07 |
| y | —— | —— | 0.08 |
| ZD | —— | 1.0 | —— |
| ZE | —— | 1.0 | —— |
| L | 0.35 | 0.5 | 0.65 |
| L1 | —— | 1.0 | —— |

**Figure 36.1 LQFP outline**

### 36.2.2 FBGA (156pin) Package Drawing

- **Figure 36.3** shows FBGA (156pin) outline.

| JEITA Package code | RENESAS code | MASS(TYP.)[g] |
|---|---|---|
| P-LFBGA156-10x10-0.65 | PLBG0156JB-A | 0.37 |




| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| D | 9.92 | 10.00 | 10.08 |
| E | 9.92 | 10.00 | 10.08 |
| A | — | — | 1.70 |
| A1 | 0.21 | 0.26 | 0.31 |
| e | — | 0.65 | — |
| b | 0.40 | 0.45 | 0.50 |
| x1 | — | — | 0.15 |
| x2 | — | — | 0.08 |
| y | — | — | 0.10 |
| y1 | — | — | 0.20 |
| n | — | 156 | — |
| ZD | — | 0.775 | — |
| ZE | — | 0.775 | — |

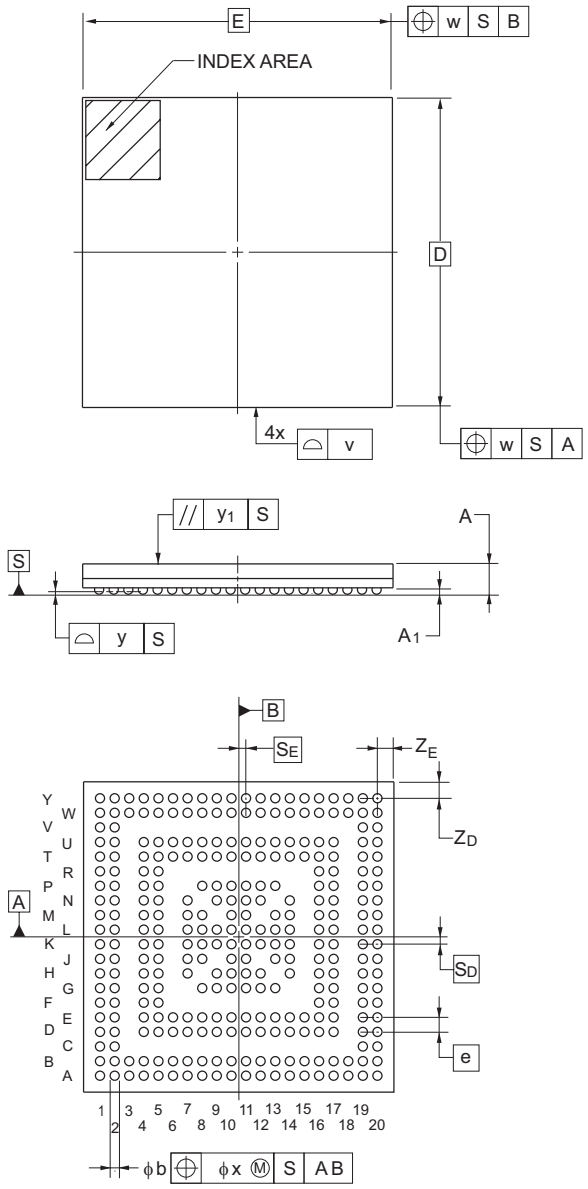
Figure 36.3 FBGA (156pin) outline

### 36.2.3 FBGA (292pin) Package Drawing

- **Figure 36.4** shows FBGA (292pin) outline.

| JEITA Package Code | RENESAS Code | Previous Code | MASS (Typ) [g] |
|---|---|---|---|
| P-FBGA292-17x17-0.80 | PRBG0292GC-A | T292F1-80-GNP | — |

Unit: mm



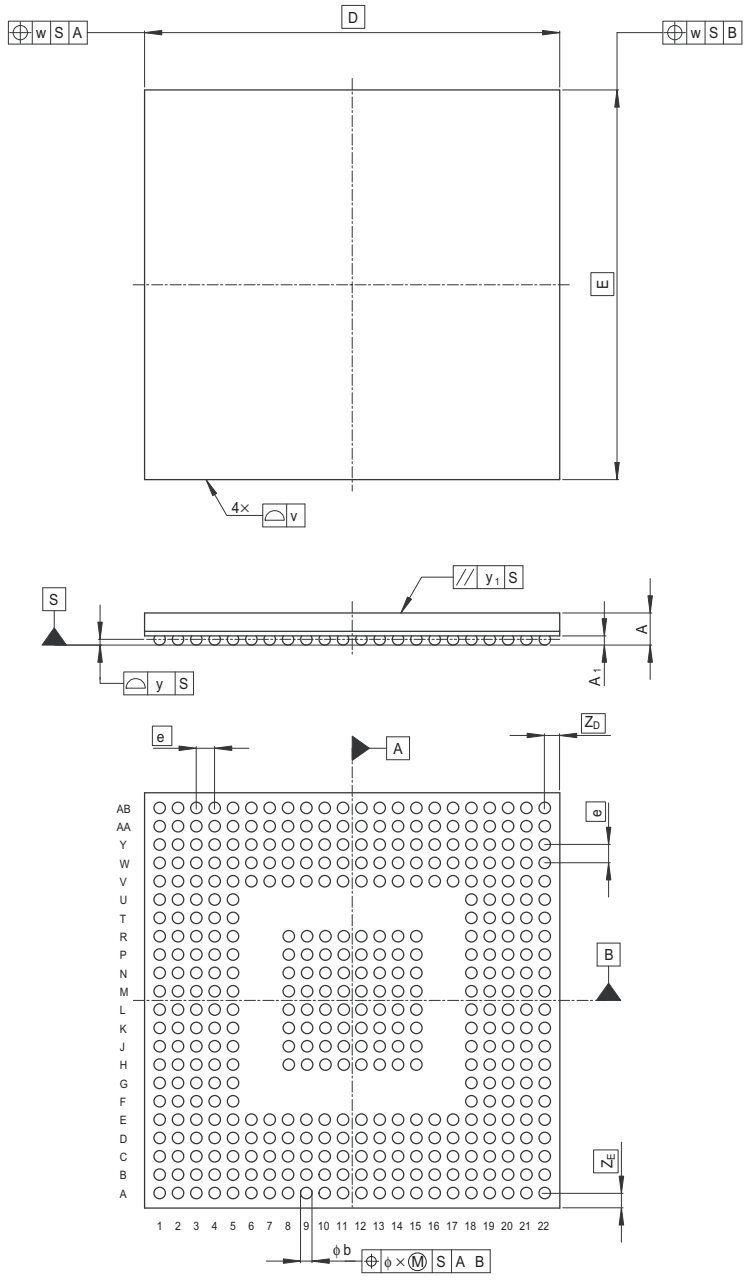

| Reference Symbol | Dimensions in millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | — | 17.00 | — |
| E | — | 17.00 | — |
| A | — | — | 1.90 |
| $A_1$ | 0.30 | 0.35 | 0.40 |
| e | — | 0.80 | — |
| b | 0.50 | 0.55 | 0.60 |
| x | — | — | 0.12 |
| y | — | — | 0.10 |
| $y_1$ | — | — | 0.20 |
| $Z_D$ | — | 0.90 | — |
| $Z_E$ | — | 0.90 | — |
| v | — | — | 0.15 |
| w | — | — | 0.20 |
| $S_D$ | — | 0.40 | — |
| $S_E$ | — | 0.40 | — |

**Figure 36.4 FBGA (292pin) outline**

### 36.2.4 FBGA (404pin) Package Drawing

- **Figure 36.5** shows FBGA (404pin) outline.

| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-FBGA404-19x19-0.80 | PRBG0404GA-A | — | 1.1g |




| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | — | 19 | — |
| E | — | 19 | — |
| v | — | — | 0.15 |
| w | — | — | 0.2 |
| A | — | — | 1.9 |
| $A_1$ | 0.35 | 0.4 | 0.45 |
| e | — | 0.8 | — |
| b | 0.45 | 0.5 | 0.55 |
| x | — | — | 0.08 |
| y | — | — | 0.1 |
| $y_1$ | — | — | 0.2 |
| $S_D$ | — | — | — |
| $S_E$ | — | — | — |
| $Z_D$ | — | 1.1 | — |
| $Z_E$ | — | 1.1 | — |

Figure 36.5 FBGA (404pin) outline

## 36.3 Differences among products

Refer **Section 36.1, Package Line Up**

# Section 37 CAN Controller (MCAN)

The Features described in this chapter are valid for all P1x-C devices except:

- R7F701370B
- R7F701371
- R7F701372A
- R7F701396A
- R7F701373A
- R7F701397A
- R7F701374A

For the MCAN description of these devices, please refer to **Section 19, CAN Controller (MCAN) According to ISO 11898-1:2015**

## 37.1 Features

The CAN controller (M(TT)CAN) is used to communicate via the CAN bus, as define in ISO11898-1. It can also support CAN FD (CAN with Flexible Data-rate) as specified in the Bosch CAN FD Specification V1.0. It covers the functionality of the Data Link Layer (DLL) and Medium Access Control (MAC). To enable the hardware to communicate, the physical layer must be provided externally, i.e., by connecting a CAN transceiver.

Bosch CAN FD Specification V1.0

**Table 37.1 M_CAN Specifications (1/2)**

| Item | Specification |
|---|---|
| Communication | CAN functionality conform to ISO 11898-1 |
| Data transfer rate | Up to 1 Mbps, individually for each CAN channel for CAN FD, up to 8 Mbps |
| CAN channels | MTTCAN0, MCAN0, MCAN1, MCAN2 |
| Selectable ID type | 11-bit Standard ID |
| | 11-bit Standard ID + 18-bit Extended ID |
| Message Buffer | Up to 64 dedicated Receive Buffers |
| | Up to 32 dedicated Transmit Buffers |
| FIFO number | Two configurable Receive FIFOs<br>Configurable Transmit FIFO<br>Configurable Transmit Queue<br>Configurable Transmit Event FIFO |
| RX System | Scalable RX FIFO structures, with up to 64 CAN Buffers per FIFO<br>RX timestamp<br>RX FIFO Timeout Interrupt<br>FIFO filling level Interrupt |
| TX System | FIFO filling level supervision (interrupt)<br>Support for transmit cancellation to avoid<br>“Inner priority inversion”<br>Combined Message Buffer & TX FIFO and TX Queue Concept<br>Dedicated TX message buffers for high-priority messages<br>ID prioritization between TX buffers, TX Queue buffers and oldest TX FIFO element<br>Transmit pause to separate two consecutive TX messages |

Table 37.1 M_CAN Specifications (2/2)

| Item | Specification |
|---|---|
| Enhanced reception filtering | Support of 11bit and 29bit CAN identifier, each filter element is configurable for acceptance/ rejection |
| | Programmable 29 bit CAN identifier acceptance filter mask for each entry |
| | Each acceptance filter element targets FIFO 0 or 1 or a dedicated RX Buffer |
| | Every FIFO or RX Buffer filter element can be used as a from-to range filter, as a filter for one or two dedicated IDs or as a classic bit mask filter |
| | Each filter element can be enabled/disabled individually |
| TT-CAN support | MTTCAN0 supports TT-CAN level 2 according to ISO11898-4 |
| CAN-FD support | Variable Data Phase Speed up to 8 Mbit/s, variable Data Buffer size up to 64 Bytes/Frame |
| Timer | Time Stamp function |
| Power down function | Local Power Down modes |
| AUTOSAR requirements | Supports all AUTOSAR requirements<br>Like Transmit Abort Interrupt, non-waiting processing<br>Functionality and more than 2 TX Buffers with prioritization<br>Supports *Pretended Networking* of AUTOSAR |
| Self-testing | External and internal loop back |

### 37.1.1 Number of Units and Channels

Table 37.2 Number of MTTCAN/MCAN channel

| | Device | | | | |
|---|---|---|---|---|---|
| Macro | P1M-C | P1H-C (4MB) | P1H-C (8MB) | P1H-CE | Description |
| MTTCAN0 | 1*[1] | 1 | 1 | 1 | MTTCAN0 supports TT-CAN level 2 according to ISO11898-4 |
| MCAN0 | 1 | 1 | 1 | 1 | — |
| MCAN1 | 1 | 1 | 1 | 1 | — |
| MCAN2 | 0 | 0 | 1 | 1 | — |

Note 1. P1M-C does not support MTTCAN0EVT, MTTCAN0RTP, MTTCAN0SOC, MTTCAN0SWT, and MTTCAN0TMP.

### 37.1.2 Register Base Address

MTTCAN/MCAN base addresses are listed in the following table. MTTCAN/MCAN register addresses are given as offsets from the base addresses in general.

Table 37.3 Register Base Address

| Base Address Name | Base Address |
|---|---|
| <MTTCAN0_base> | FFD3 $0000_H$ |
| <MCAN0_base> | FFEF $0000_H$ |
| <MCAN1_base> | FFD3 $1000_H$ |
| <MCAN2_base> | FFEF $1000_H$ |

### 37.1.3 Clock Supply

Clock supply by and to MTTCAN/MCAN is listed in the following table.

**Table 37.4 Clock Supply**

| Unit Name | Clock for the Unit | Supply Clock Name |
|---|---|---|
| MTTCAN0/ MCANn | H-Bus interface clock*1 (Host clock) | CLK_HSB |
| | CAN protocol layer clock*2 (CAN clock / m_can_cclk/ m_ttcan_cclk) | CLKP_H2 |

**Note:** The following condition must be fulfilled: CLK_HSB >= CLKP_H2

Note 1. H-bus interface clock:
The minimum frequency is **Table 37.5, Minimune MCAN / MTTCAN clock speed**.

Note 2. CAN protocol layer clock
- For maximum data rate, CLKP_H2 must be n x 1 MHz with n >= 8.
- For CAN-FD operation with bit rates above 1Mbit/s CLKP_H2 must be 2^n x 20MHz, with n = 0, 1, 2.
- For low power mode with limited throughput it can be reduce to CLK_HSB/3. This limits the throughput to 333Kbps.

**Table 37.5 Minimune MCAN / MTTCAN clock speed**

| | P1M-C | P1H-C (4MB) | P1H-C (8MB) |
|---|---|---|---|
| Number of channels | 3 | 3 | 4 |
| Non CANFD | 17.5 MHz | | |
| CANFD | | | |

Bosch CAN-FD Specification V1.0

### 37.1.4 RAM areas for MCAN

**Table 37.6 RAM areas for MCAN**

| RAM Name | RAM Area |
|---|---|
| MTTCAN RAM area | FFD3 $8000_H$ to FFD3 $9FFF_H$ |
| MCAN0 RAM area | FFEF $8000_H$ to FFEF $9FFF_H$ |
| MCAN1 RAM area | FFD3 $A000_H$ to FFD3 $BFFF_H$ |
| MCAN2 RAM area | FFEF $A000_H$ to FFEF $BFFF_H$ |

### 37.1.5 Interrupt Requests

MTTCAN/MCAN interrupt requests are listed in the following table.

Table 37.7 Interrupt list

| Unit Name | Interrupt Name | Interrupt Number | DMA Number | Description |
|---|---|---|---|---|
| MTTCAN0 | INTMTTCANI0 | 172 | — | MTTCAN0 interrupt 0 |
| MTTCAN0 | INTMTTCANI1 | 173 | — | MTTCAN0 interrupt 1 |
| MTTCAN0 | INTMTTCANFE | 174 | — | MTTCAN0 filter event 1 |
| MCAN0 | INTMCAN0I0 | 175 | — | MCAN0 interrupt 0 |
| MCAN0 | INTMCAN0I1 | 176 | — | MCAN0 interrupt 1 |
| MCAN0 | INTMCAN0FE | 177 | — | MCAN0 filter event 1 |
| MCAN1 | INTMCAN1I0 | 178 | — | MCAN1 interrupt 0 |
| MCAN1 | INTMCAN1I1 | 179 | — | MCAN1 interrupt 1 |
| MCAN1 | INTMCAN1FE | 180 | — | MCAN1 filter event 1 |
| MCAN2 | INTMCAN2I0 | 181 | — | MCAN2 interrupt 0 |
| MCAN2 | INTMCAN2I1 | 182 | — | MCAN2 interrupt 1 |
| MCAN2 | INTMCAN2FE | 183 | — | MCAN2 filter event 1 |

Table 37.8 Internal Error Signal

| Unit Name | Interrupt for Unit | Description | Connected to |
|---|---|---|---|
| MTTCAN0 | mttcan0_ecc_err | MTTCAN0 error signal | ECM |
| MCAN0 | mcan0_ecc_err | MCAN0 error signal | ECM |
| MCAN1 | mcan1_ecc_err | MCAN1 error signal | ECM |
| MCAN2 | mcan2_ecc_err | MCAN2 error signal | ECM |

Bosch CAN-FD Specification V1.0

### 37.1.6 External Input / Output Pins

MTTCAN/MCAN has following external pins for each channel.

Table 37.9 External Input/Output Pins

| Channel | I/O | Pin name of RH850/P1x-C | Function |
|---|---|---|---|
| MTTCAN0 | I | MTTCAN0RX | MTTCAN0 receive data input |
| | O | MTTCAN0TX | MTTCAN0 transmit data output |
| | I | MTTCAN0SWT | MTTCAN0 stop watch trigger |
| | I | MTTCAN0EVT | MTTCAN0 event trigger |
| | O | MTTCAN0RTP | MTTCAN0 register time mark interrupt pulse |
| | O | MTTCAN0TMP | MTTCAN0 trigger time mark interrupt pulse |
| | O | MTTCAN0SOC | MTTCAN0 start of cycle |
| MCAN0 | I | MCAN0RX | MCAN0 receive data input |
| | O | MCAN0TX | MCAN0 transmit data output |
| MCAN1 | I | MCAN1RX | MCAN1 receive data input |
| | O | MCAN1TX | MCAN1 transmit data output |
| MCAN2 | I | MCAN2RX | MCAN2 receive data input |
| | O | MCAN2TX | MCAN2 transmit data output |
| MCKDV | O | BHPDGRCLK0 | Degrading clock output for MCAN |

## 37.2 Overview

### 37.2.1 Functional overview

M(TT)CAN has following features

- M(TT)CAN functionality conform to ISO 11898-1
- M(TT)CAN supports local Power Down modes
- M(TT)CAN supports all AUTOSAR requirements
  - Transmit Abort Interrupt
  - Non-waiting processing functionality
  - Include more than 2 TX Buffers prioritization
- M(TT)CAN supports several measures for self-testing:external and internal loop back
  - For this reason, the port structure does not have additional functionality for self-testing
  - The usage of loop back functionality for safety requirements is described in **Section 28, Functional Safety**
- M(TT)CAN contains an improved RX System
  - Scalable RX FIFO structures, with up to 64 CAN Buffers per FIFO
  - RX timestamp
  - RX FIFO Timeout Interrupt
  - FIFO filling level Interrupt
- M(TT)CAN contains an improved TX System
  - Variable amount of 0 to 32 “classical” TX Buffers
  - Additional, size configurable TX FIFO
  - Additional, size configurable TX Queue
  - FIFO filling level supervision (interrupt)
  - Support for transmit cancellation to avoid “inner priority inversion”
  - Configurable TX Event history: For each transmitted message both ID and timestamp are written into a history
  - TX-Event FIFO (readable by CPU using polling or interrupts) containing 0 to 32 events
  - Combined Message Buffer & TX FIFO and TX Queue Concept
  - Dedicated TX message buffers for high-priority messages
  - ID prioritization between TX buffers, TX Queue buffers and oldest TX FIFO element
  - Transmit pause to separate two consecutive TX messages
- M(TT)CAN contains an enhanced reception filtering
  - Support of 11bit and 29bit CAN identifier, each filter element is configurable for acceptance/ rejection
  - Each acceptance filter element targets FIFO 0 or 1 or a dedicated RX Buffer
  - Every FIFO or RX Buffer filter element can be used as a from-to range filter, as a filter for one or two dedicated
  - IDs or as a classic bit mask filter

Bosch CAN-FD Specification V1.0

  - Each filter element can be enabled/disabled individually
  - Programmable 29 bit CAN identifier acceptance filter mask for each entry
- MTTCAN supports TT_CAN level 2 according to ISO11898-4
- MTTCAN supports additional synchronization input signals
  - MTTCAN supports additional status output signals
- Data transfer rate is up to 1Mbps, individually for each CAN channel. For CAN FD, up to 8Mbps
- M(TT)CAN supports Pretended Networking of AUTOSAR
  - This can be achieved by using the dedicated additional interrupt assigned to messages of filter group 1
- Selectable ID type
  - 11-bit Standard ID
  - 11-bit Standard ID + 18-bit Extended ID
- M(TT)CAN are supported by the debug system by means of a debug stop signal, which suppresses the register content modification by reading of accesses of the debugger
- M(TT)CAN supports Time Stamp function
- M(TT)CAN are supporting CAN-FD with 64 data bytes and flexible data rate

Bosch CAN FD Specification V1.0

### 37.2.2 Block Diagram

Following block diagram shows MTTCAN/MCAN block diagram. P1H-C (8MB) and P1H-CE support MCAN2. P1M-C and P1H-C (4MB) does not support MCAN2.

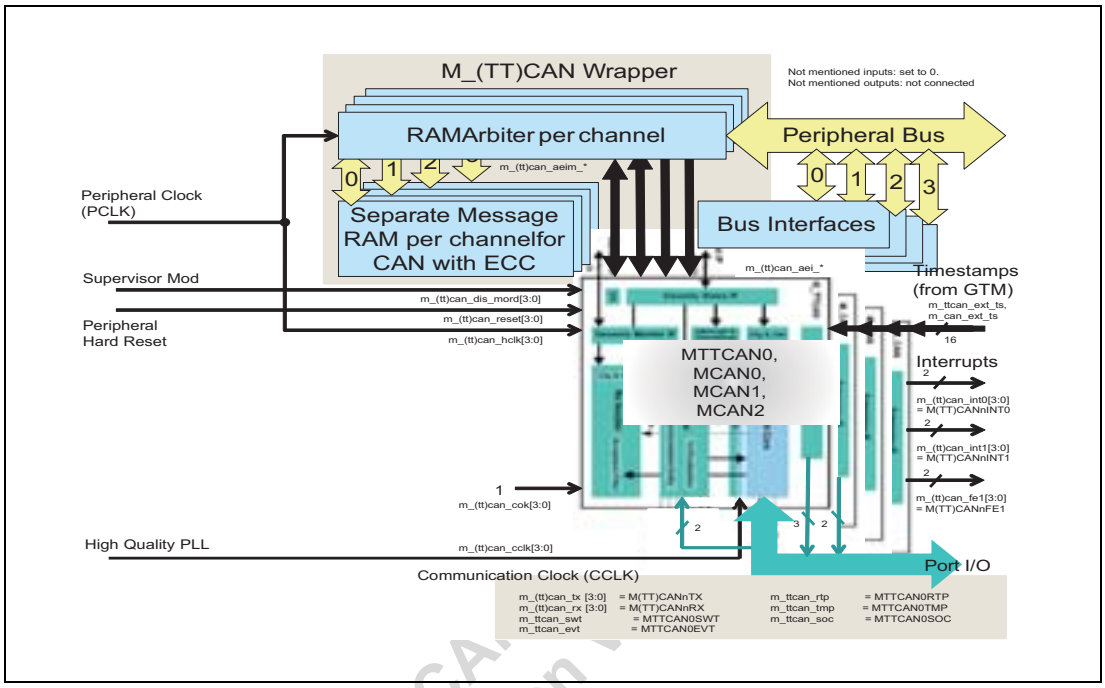



Figure 37.1 MCAN/MTTCAN block diagram

- M_TTCAN (MTTCAN0)
  8K RAM for messages, events and filters; no CCU implemented
  additional input signals MTTCAN0SWT, MTTCAN0EVT;
  additional output signals MTTCAN0RTP, MTTCAN0TMP, MTTCAN0SOC;
  timestamps from GTM IP
- M_CAN (MCAN0)
  8K RAM for messages, events and filters; common clocking with MTTCAN0
  timestamps from GTM IP
- M_CAN (MCAN1)
  8K RAM for messages, events and filters; common clocking with MTTCAN0
  timestamps from GTM IP
- M_CAN (MCAN2)
  8K RAM for messages, events and filters; common clocking with MTTCAN0
  timestamps from GTM IP
  P1M-C and P1H-C (4MB) don't support MCAN2.

  *m_(tt) can_cok fixed "1"*1

**Note 1.** This signal of M_(TT) CAN indicates that a proper clock supply is available. In devices P1M-C and P1H-C (incl. P1H-CE), clock is provided permanently so that M_(TT) CAN is enabled for transmission by this signal without precondition.

# 37.3 Registers

## 37.3.1 List of Registers

MTTCAN/MCAN registers are listed in the following table.

**Table 37.10 Register list**

| Address | Register name | Description | Access Size[bit] | Initial Value | Access Protection | |
|---|---|---|---|---|---|---|
| | | | | | PBG | Other |
| *1 | *1 | MTTCAN0 | 32 | *1 | PBG3#1.PG3-MCANT | — |
| *1 | *1 | MCAN0 | 32 | *1 | PBG1#1.PG1-MCAN0 | — |
| *1 | *1 | MCAN1 | 32 | *1 | PBG3#1.PG3-MCAN1 | — |
| *1 | *1 | MCAN2 | 32 | *1 | PBG1#1.PG1-MCAN2 | — |

Note 1. For details of MTTCAN/MCAN-IP registers, please see **Table 37.68, M_TTCAN Register Map** and **Table 37.12, M_CAN Register Map**.

**Table 37.11 Register Reset Condition**

| Register Name | Reset condition | | | | |
|---|---|---|---|---|---|
| | Power On Reset | System Reset1 | System Reset2 | Application Reset1 | Limited Reset |
| All registers | √ | √ | √ | √ | √ |

Bosch CAN-FD Specification V1.

## 37.4 Operation

### 37.4.1 Procedure of Module Standby and Limited Reset

This module supports module standby and limited reset functions. Before these functions are enabled, all of the followings must be ensured:

- Module standby
  The M(TT)CAN can be set into power down mode controlled by CC Control Register M(TT)CANnCCCR.CSR. As long as the clock stop request is active, bit M(TT)CANnCCCR.CSR is read as one. When all pending transmission requests have completed, the M(TT)CAN waits until bus idle state is detected. Then the M(TT)CAN sets M(TT)CANnCCCR.INIT to one to prevent any further CAN transfers. Now the M(TT)CAN acknowledges that it is ready for power down by setting M(TT)CANnCCCR.CSA to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to M(TT)CANnCCCR.INIT will have no effect. Now the module clock may be switched off. To leave power down mode, the application has to turn on the module clocks before resetting the CC Control Register flag M(TT)CANnCCCR.CSR. The M_CAN will acknowledge this by resetting M(TT)CANnCCCR.CSA. Afterwards, the application can restart CAN communication by resetting bit M(TT)CANnCCCR.INIT. Refer to the following flow chart.
- Limited Reset
  The M(TT)CAN can be reset by limited reset from SYSCTRL. The Limited Reset is functionally equivalent to a hardware reset. After limited reset, the registers of the M(TT)CAN hold the reset values. Additionally the Bus_Off state is reset and the output “m_(tt)can_tx” is set to recessive (HIGH). The value $0001_H$ (M(TT)CANnCCCR.INIT = ‘1’) in the CC Control Register enables software initialization. The M(TT)CAN does not influence the CAN bus until the CPU resets M(TT)CANnCCCR.INIT to ‘0’. Refer to the following flow chart.

Bosch CAN-FD Specification V1.0

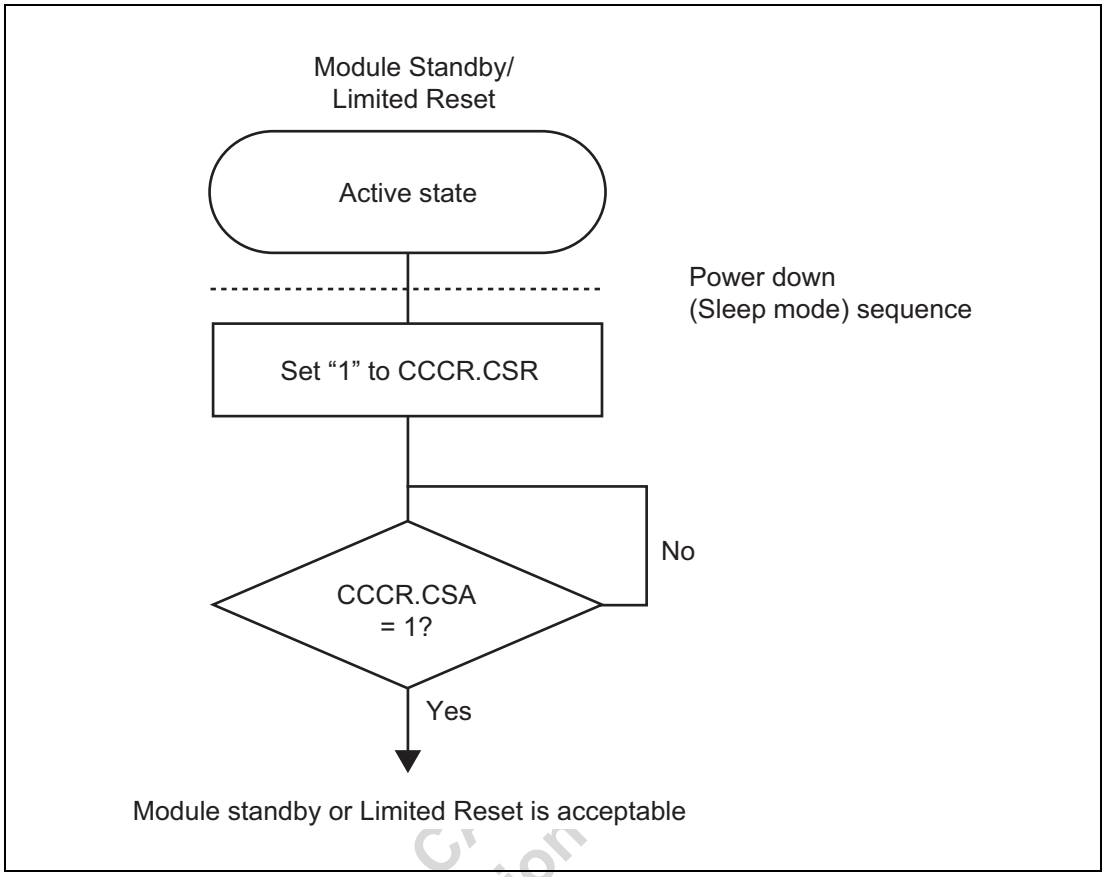



**Figure 37.2 Module Standby/Limited Reset procedure**

Bosch CAN Specification

# 37.5 MCAN

## 37.5.1 Overview

The M_CAN performs communication according to ISO11898-1 (Bosch CAN specification 2.0 part A, B) and to Bosch CAN FD specification V1.0. Additional transceiver hardware is required for connection to the physical layer.

The message storage is intended to be a single-ported Message RAM outside of the module. It is connected to the M_CAN via the Generic Master Interface.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN Core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN Core as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

### 37.5.1.1 Features

- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1
- CAN FD with up to 64 data bytes supported
- CAN Error Logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signalling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains (CAN clock and Host clock)
- Power-down support

Bosch CAN FD Specification V1.0

### 37.5.1.2 Block Diagram

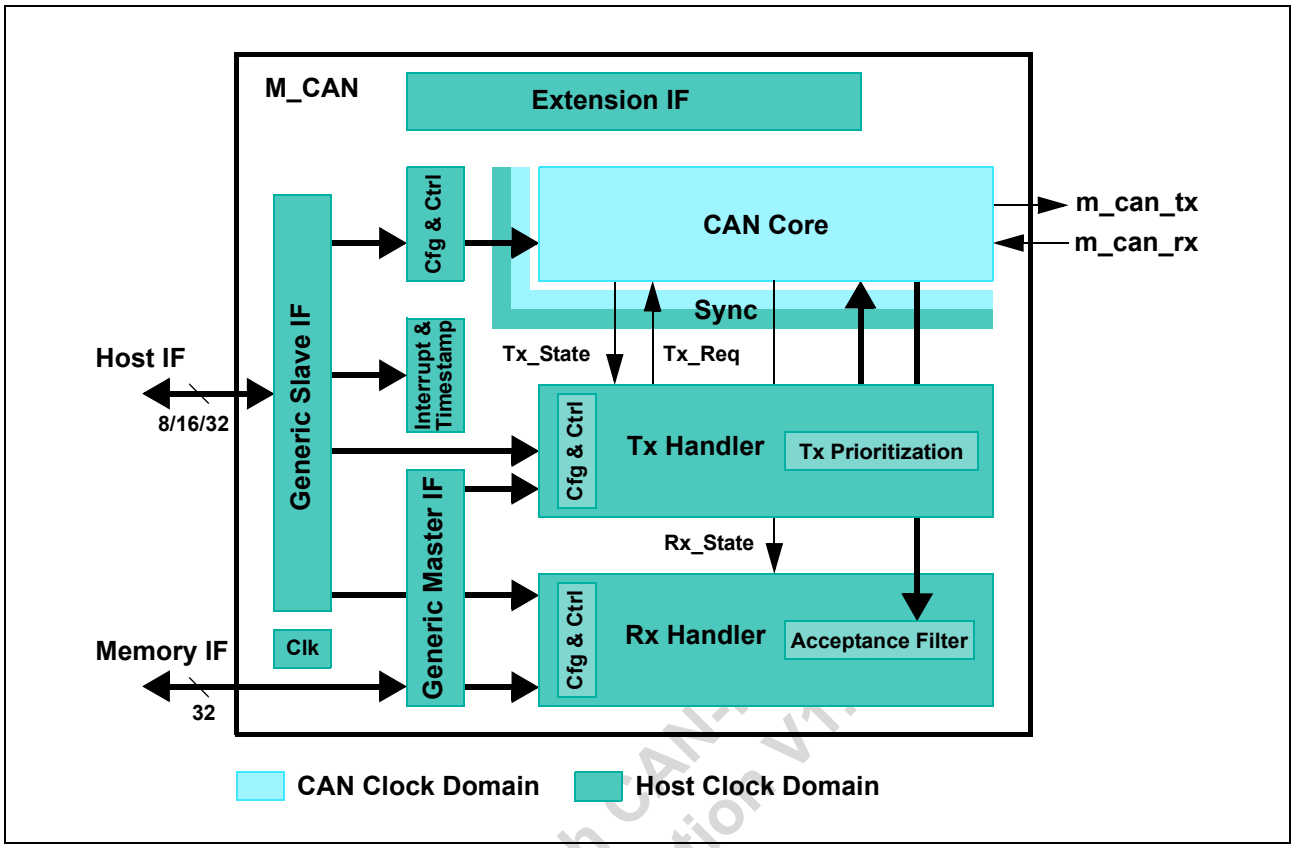



Figure 37.3 M_TTCAN Block Diagram

**CAN Core:**

CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.

**Sync:**

Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.

**Clk:**

Synchronizes reset signal to the Host clock domain and to the CAN clock domain.

**Cfg & Ctrl:**

CAN Core related configuration and control bits.

**Interrupt & Timestamp:**

Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation. An externally generated 16-bit vector may substitute the integrated 16-bit CAN bit time counter for receive and transmit timestamp generation.

**Tx Handler:**

Controls the message transfer from the external Message RAM to the CAN Core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.

**Rx Handler:**

Controls the transfer of received messages from the CAN Core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

**Generic Slave Interface:**

Connects the M_CAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8/16/32-bit bus to support a wide range of interconnection structures.

**Generic Master Interface:**

Connects the M_CAN to a local 32-bit Message RAM. The implemented Message RAM size is 2K • 32 bit.

Extension Interface:

All flags from the Interrupt Register MCANnIR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M_CAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

Bosch CAN FD Specification V1.0

### 37.5.1.3 Dual Clock Sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain m_can_hclk (CLK_HSB). Due to the high precision clocking requirements of the CAN Core, a separate clock without any modulation has to be provided as m_can_cclk (CLKP_H2).

Within the M_TTCAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

**NOTE**

In order to achieve a stable function of the M_TTCAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

### 37.5.1.4 Dual Interrupt Lines

The module provides two interrupt lines. Interrupts can be routed either to m_can_int0 (INTMCANnI0) or to m_can_int1 (INTMCANnI1). By default all interrupts are routed to interrupt line m_can_int0 (INTMCANnI0). By programming MCANnILE.EINT0 and MCANnILE.EINT1 the interrupt lines can be enabled or disabled separately.

## 37.5.2 Programmer's Model

### 37.5.2.1 Hardware Reset Description

After hardware reset, the registers of the M_TTCAN hold the reset values listed in **Table 37.12**. Additionally the Bus_Off state is reset and the output m_can_tx is set to recessive (HIGH). The value $0001_H$ (MCANnCCCR.INIT = '1') in the CC Control Register enables software initialization. The M_TTCAN does not influence the CAN bus until the CPU resets MCANnCCCR.INIT to '0'.

### 37.5.2.2 Register Map

The M_TTCAN module allocates an address space of 256 bytes. All registers are organized as 32-bit registers. The M_CAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8 bit (byte access), 16 bit (half-word access), or 32 bit (word access). Write access by the Host CPU to registers/bits marked with "P = Protected Write" is possible only with MCANnCCCR.CCE = '1' AND MCANnCCCR.INIT = '1'. There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

**Table 37.12 M_CAN Register Map (1/2)**

| ADDRESS | SYMBOL | NAME | page | RESET | ACC |
|---|---|---|---|---|---|
| <MCANn_base> + $000_H$ | MCANnCREL | Core Release Register | 2831 | 3013 0506 | R |
| <MCANn_base> + $004_H$ | MCANnENDN | Endian Register | 2832 | 8765 4321 | R |
| <MCANn_base> + $00C_H$ | MCANnFBTP | Data Bit Timing & Prescaler Register | 2833 | 0000 0A33 | RP |
| <MCANn_base> + $010_H$ | MCANnTEST | Test Register | 2835 | 0000 0000 | RP |
| <MCANn_base> + $014_H$ | MCANnRWD | RAM Watchdog | 2836 | 0000 0000 | RP |
| <MCANn_base> + $018_H$ | MCANnCCCR | CC Control Register | 2837 | 0000 0001 | RWPp |
| <MCANn_base> + $01C_H$ | MCANnBTP | Bit Timing & Prescaler Register | 2839 | 0000 0A33 | RP |
| <MCANn_base> + $020_H$ | MCANnTSCC | Timestamp Counter Configuration | 2840 | 0000 0000 | RP |
| <MCANn_base> + $024_H$ | MCANnTSCV | Timestamp Counter Value | 2841 | 0000 0000 | RC |
| <MCANn_base> + $028_H$ | MCANnTOCC | Timeout Counter Configuration | 2842 | FFFF 0000 | RP |
| <MCANn_base> + $02C_H$ | MCANnTOCV | Timeout Counter Value | 2843 | 0000 FFFF | RC |
| <MCANn_base> + $030_H$ to $03C_H$ | | reserved (4) | | 0000 0000 | R |
| <MCANn_base> + $040_H$ | MCANnECR | Error Counter Register | 2844 | 0000 0000 | RX |
| <MCANn_base> + $044_H$ | MCANnPSR | Protocol Status Register | 2845 | 0000 0707 | RXS |
| <MCANn_base> + $048_H$ to $04C_H$ | | reserved (2) | | 0000 0000 | R |
| <MCANn_base> + $050_H$ | MCANnIR | Interrupt Register | 2847 | 0000 0000 | RW |
| <MCANn_base> + $054_H$ | MCANnIE | Interrupt Enable | 2850 | 0000 0000 | RW |
| <MCANn_base> + $058_H$ | MCANnILS | Interrupt Line Select | 2852 | 0000 0000 | RW |
| <MCANn_base> + $05C_H$ | MCANnILE | Interrupt Line Enable | 2854 | 0000 0000 | RW |
| <MCANn_base> + $060_H$ to $07C_H$ | | reserved (8) | | 0000 0000 | R |
| <MCANn_base> + $080_H$ | MCANnGFC | Global Filter Configuration | 2855 | 0000 0000 | RP |
| <MCANn_base> + $084_H$ | MCANnSIDFC | Standard ID Filter Configuration | 2856 | 0000 0000 | RP |
| <MCANn_base> + $088_H$ | MCANnXIDFC | Extended ID Filter Configuration | 2857 | 0000 0000 | RP |
| <MCANn_base> + $08C_H$ | | reserved (1) | | 0000 0000 | R |
| <MCANn_base> + $090_H$ | MCANnXIDAM | Extended ID AND Mask | 2858 | 1FFF FFFF | RP |
| <MCANn_base> + $094_H$ | MCANnHPMS | High Priority Message Status | 2859 | 0000 0000 | R |
| <MCANn_base> + $098_H$ | MCANnNDAT1 | New Data 1 | 2860 | 0000 0000 | RW |
| <MCANn_base> + $09C_H$ | MCANnNDAT2 | New Data 2 | 2861 | 0000 0000 | RW |
| <MCANn_base> + $0A0_H$ | MCANnRXF0C | Rx FIFO 0 Configuration | 2862 | 0000 0000 | RP |

Bosch CAN FD Specification V1.0

Table 37.12 M_CAN Register Map (2/2)

| ADDRESS | SYMBOL | NAME | page | RESET | ACC |
|---|---|---|---|---|---|
| <MCANn_base> + $0A4_H$ | MCANnRXF0S | Rx FIFO 0 Status | 2863 | 0000 0000 | R |
| <MCANn_base> + $0A8_H$ | MCANnRXF0A | Rx FIFO 0 Acknowledge | 2864 | 0000 0000 | RW |
| <MCANn_base> + $0AC_H$ | MCANnRXBC | Rx Buffer Configuration | 2865 | 0000 0000 | RP |
| <MCANn_base> + $0B0_H$ | MCANnRXF1C | Rx FIFO 1 Configuration | 2866 | 0000 0000 | RP |
| <MCANn_base> + $0B4_H$ | MCANnRXF1S | Rx FIFO 1 Status | 2867 | 0000 0000 | R |
| <MCANn_base> + $0B8_H$ | MCANnRXF1A | Rx FIFO 1 Acknowledge | 2868 | 0000 0000 | RW |
| <MCANn_base> + $0BC_H$ | MCANnRXESC | Rx Buffer / FIFO Element Size Configuration | 2869 | 0000 0000 | RP |
| <MCANn_base> + $0C0_H$ | MCANnTXBC | Tx Buffer Configuration | 2870 | 0000 0000 | RP |
| <MCANn_base> + $0C4_H$ | MCANnTXFQS | Tx FIFO/Queue Status | 2871 | 0000 0000 | R |
| <MCANn_base> + $0C8_H$ | MCANnTXESC | Tx Buffer Element Size Configuration | 2872 | 0000 0000 | RP |
| <MCANn_base> + $0CC_H$ | MCANnTXBRP | Tx Buffer Request Pending | 2873 | 0000 0000 | R |
| <MCANn_base> + $0D0_H$ | MCANnTXBAR | Tx Buffer Add Request | 2874 | 0000 0000 | RW |
| <MCANn_base> + $0D4_H$ | MCANnTXBCR | Tx Buffer Cancellation Request | 2875 | 0000 0000 | RW |
| <MCANn_base> + $0D8_H$ | MCANnTXBTO | Tx Buffer Transmission Occurred | 2876 | 0000 0000 | R |
| <MCANn_base> + $0DC_H$ | MCANnTXBCF | Tx Buffer Cancellation Finished | 2877 | 0000 0000 | R |
| <MCANn_base> + $0E0_H$ | MCANnTXBTIE | Tx Buffer Transmission Interrupt Enable | 2877 | 0000 0000 | RW |
| <MCANn_base> + $0E4_H$ | MCANnTXBCIE | Tx Buffer Cancellation Finished Interrupt Enable | 2878 | 0000 0000 | RW |
| <MCANn_base> + $0E8_H$ to $0EC_H$ | | reserved (2) | | 0000 0000 | R |
| <MCANn_base> + $0F0_H$ | MCANnTXEFC | Tx Event FIFO Configuration | 2879 | 0000 0000 | RP |
| <MCANn_base> + $0F4_H$ | MCANnTXEFS | Tx Event FIFO Status | 2880 | 0000 0000 | R |
| <MCANn_base> + $0F8_H$ | MCANnTXEFA | Tx Event FIFO Acknowledge | 2881 | 0000 0000 | RW |
| <MCANn_base> + $0FC_H$ | | reserved (1) | | 0000 0000 | R |

**Note:** R = Read, S = Set on read, X = Reset on read, W = Write, P = Protected write, p = Protected set, C = Clear/preset on write

Bosch CAN-FD Specification V1.0

### 37.5.2.3 Registers

#### (1) MCANnCREL — Core Release Register

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $000_H$

**Value after reset:** $3013\ 0506_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | REL[3:0] | | | | STEP[3:0] | | | | SUBSTEP[3:0] | | | | YEAR[3:0] | | | |
| Value after reset | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MON[7:0] | | | | | | | | DAY[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.13 MCANnCREL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | REL[3:0] | Core Release<br>One digit, BCD-coded. |
| 27 to 24 | STEP[3:0] | Step of Core Release<br>One digit, BCD-coded. |
| 23 to 20 | SUBSTEP[3:0] | Sub-step of Core Release<br>One digit, BCD-coded. |
| 19 to 16 | YEAR[3:0] | Time Stamp Year<br>One digit, BCD-coded. This field is set by generic parameter on M_CAN synthesis. |
| 15 to 8 | MON[7:0] | Time Stamp Month<br>Two digits, BCD-coded. This field is set by generic parameter on M_CAN synthesis. |
| 7 to 0 | DAY[7:0] | Time Stamp Day<br>Two digits, BCD-coded. This field is set by generic parameter on M_CAN synthesis. |

**Table 37.14 Coding of Revisions**

| Release | Step | SubStep | Year | Month | Day | Name |
|---|---|---|---|---|---|---|
| 3 | 0 | 1 | 3 | 05 | 06 | Revision 3.0.1, Date 2013/05/06 |

### (2) MCANnENDN — Endian Register

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $004_H$

**Value after reset:** $8765\ 4321_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETV[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETV[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.15 MCANnENDN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ETV[31:0] | Endianness Test Value<br>The endianness test value is $87654321_H$. |

Bosch CAN-FD Specification V1.0

### (3) MCANnFBTP — Fast Bit Timing & Prescaler Register

This register is only writable if bits MCANnCCCR.CCE and MCANnCCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 m_can_cclk (CLKP_H2) periods. tq = (FBRP + 1) m_can_cclk (CLKP_H2) period.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. FTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [FTSEG1 + FTSEG2 + 3] tq
or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $00C_H$

**Value after reset:** 0000 $0A33_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | TDCO[4:0] | | | | | TDC | — | — | FBRP[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | RP | RP | RP | RP | RP | RP | R | R | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | FTSEG1[3:0] | | | | — | FTSEG2[2:0] | | | — | — | FSJW[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | RP | RP | RP | RP | R | RP | RP | RP | R | R | RP | RP |

Bosch CAN-FD Specification V1.0

**Table 37.16 MCANnFBTP Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. |
| 28 to 24 | TDCO[4:0] | Transmitter Delay Compensation Offset<br>$00_H$ to $1F_H$<br>Offset value defining the distance between the measured delay from m_can_tx to m_can_rx and the secondary sample point. Valid values are 0 to 31 m_can_cclk (CLKP_H2) periods. |
| 23 | TDC | Transmitter Delay Compensation<br>0: Transmitter Delay Compensation disabled<br>1: Transmitter Delay Compensation enabled |
| 22, 21 | Reserved | These bits are always read as 0. |
| 20 to 16 | FBRP[4:0] | Fast Baud Rate Prescaler<br>$00_H$ to $1F_H$<br>The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 15 to 12 | Reserved | These bits are always read as 0. |
| 11 to 8 | FTSEG1[3:0] | Fast time segment before sample point<br>$1_H$ to $F_H$<br>Valid values are 1 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 7 | Reserved | This bit is always read as 0. |

**Table 37.16 MCANnFBTP Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 to 4 | FTSEG2[2:0] | Fast time segment after sample point<br>$0_H$ to $7_H$<br>Valid values are 0 to 7. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 3, 2 | Reserved | These bits are always read as 0. |
| 1, 0 | FSJW[1:0] | Fast (Re) Synchronization Jump Width<br>$0_H$ to $3_H$<br>Valid values are 0 to 3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |

**NOTES**

1. With a CAN clock (m_can_cclk: CLKP_H2) of 8 MHz, the reset value of 0000 $0A33_H$ configures the M_CAN for a fast bit rate of 500 kBit/s.
2. The bit rate configured for the CAN FD data phase via MCANnFBTP must be higher or equal to the bit rate configured for the arbitration phase via MCANnBTP.

Bosch CAN-FD Specification V1.0

### (4) MCANnTEST — Test Register

Write access to the Test Register has to be enabled by setting bit MCANnCCCR.TEST to '1'. All Test Register functions are set to their reset values when bit MCANnCCCR.TEST is reset.

Loop Back Mode and software control of pin m_can_tx are hardware test modes. Programming of TX ≠ "00" may disturb the message transfer on the CAN bus.

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $010_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TDCV[5:0] | | | | | | RX | TX[1:0] | | LBCK | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | RP | RP | RP | R | R | R | R |

**Table 37.17 MCANnTEST Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | These bits are always read as 0. |
| 13 to 8 | TDCV[5:0] | Transmitter Delay Compensation Value<br>$00_H$ to $3F_H$<br>Position of the secondary sample point, defined by the sum of the measured delay from m_can_tx to m_can_rx and MCANnFBTP.TDCO. Valid values are 0 to 63 m_can_cclk (CLKP_H2) periods. |
| 7 | RX | Receive Pin<br>Monitors the actual value of pin m_can_rx<br>0: The CAN bus is dominant (m_can_rx = '0').<br>1: The CAN bus is recessive (m_can_rx = '1'). |
| 6, 5 | TX[1:0] | Control of Transmit Pin<br>00: Reset value, m_can_tx controlled by the CAN Core, updated at the end of the CAN bit time.<br>01: Sample Point can be monitored at pin m_can_tx.<br>10: Dominant ('0') level at pin m_can_tx<br>11: Recessive ('1') at pin m_can_tx |
| 4 | LBCK | Loop Back Mode<br>0: Reset value, Loop Back Mode is disabled.<br>1: Loop Back Mode is enabled (see **(9) Test Modes**). |
| 3 to 0 | Reserved | These bits are always read as 0. |

Bosch M_CAN-FD Specification V1.0

### (5) MCANnRWD — RAM Watchdog

The RAM Watchdog monitors the readiness of the Message RAM. A Message RAM access via the M_CAN's Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by MCANnRWD.WDC. The counter is reloaded with MCANnRWD.WDC when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag MCANnIR.WDI is set. The RAM Watchdog Counter is clocked by the Host clock (m_can_hclk: CLK_HSB).

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $014_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WDV[7:0] | | | | | | | | WDC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP |

Bosch CAN FD Specification V1.0

**Table 37.18 MCANnRWD Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 8 | WDV[7:0] | Watchdog Value<br>Actual Message RAM Watchdog Counter Value. |
| 7 to 0 | WDC[7:0] | Watchdog Configuration<br>Start value of the Message RAM Watchdog Counter. With the reset value of "00" the counter is disabled. |

### (6) MCANnCCCR — CC Control Register

For details about setting and resetting of single bits see **(1) Software Initialization**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $018_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TXP | FDBS | FDO | CMR[1:0] | | CME[1:0] | | TEST | DAR | MON | CSR | CSA | ASM | CCE | INIT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | RP | R | R | R/W | R/W | RP | RP | Rp | RP | Rp | R/W | R | Rp | RP | R/W |

**Table 37.19 MCANnCCCR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 15 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 14 | TXP | Transmit Pause<br>If this bit is set, the M_CAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame (see **Section 37.6.3.5, Tx Handling**).<br>0: Transmit pause disabled<br>1: Transmit pause enabled |
| 13 | FDBS | CAN FD Bit Rate Switching<br>0: This node transmits no frames with bit rate switching.<br>1: This node transmits all frames (excl. remote frames) with bit rate switching. |
| 12 | FDO | CAN FD Operation<br>0: This node transmits all frames in CAN format according to ISO11898-1.<br>1: This node transmits all frames (excl. remote frames) in CAN FD format. |
| 11, 10 | CMR[1:0] | CAN Mode Request<br>A change of the CAN operation mode is requested by writing to this bit field. After change to the requested operation mode the bit field is reset to “00” and the status flags FDBS and FDO are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CMR is retained until it is overwritten by the next mode change request. In case CME = “01”/”10”/”11” a change to CAN operation according to ISO 11898-1 is always possible. Default is CAN operation according to ISO11898-1.<br>00: Unchanged<br>01: Request CAN FD operation<br>10: Request CAN FD operation with bit rate switching<br>11: Request CAN operation according ISO11898-1 |
| 9, 8 | CME[1:0] | CAN Mode Enable<br>00: CAN operation according to ISO11898-1 enabled<br>01: CAN FD operation enabled<br>10: CAN FD operation with bit rate switching enabled<br>11: CAN FD operation with bit rate switching enabled<br>**NOTE**<br>When CME = “00”, received frames are strictly interpreted according to ISO11898-1, which leads to the transmission of an error frame when receiving a CAN FD frame. In case CME = “01”, transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CME = “10”/“11”, transmission and reception of long and fast CAN FD frames is enabled. |

Boot Mode Specification

**Table 37.19 MCANnCCCR Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | TEST | Test Mode Enable<br>0: Normal operation, register MCANnTEST holds reset values.<br>1: Test Mode, write access to register MCANnTEST enabled. |
| 6 | DAR | Disable Automatic Retransmission<br>0: Automatic retransmission of messages not transmitted successfully enabled.<br>1: Automatic retransmission disabled |
| 5 | MON | Bus Monitoring Mode<br>Bit MON can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time.<br>0: Bus Monitoring Mode is disabled<br>1: Bus Monitoring Mode is enabled |
| 4 | CSR | Clock Stop Request<br>0: No clock stop is requested.<br>1: Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle. |
| 3 | CSA | Clock Stop Acknowledge<br>0: No clock stop acknowledged.<br>1: M_TTCAN may be set in power down by stopping m_can_hclk (CLK_HSB) and m_can_cclk (CLKP_H2). |
| 2 | ASM | Restricted Operation Mode<br>Bit ASM can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. For a description of the Restricted Operation Mode see **(5) Restricted Operation Mode**.<br>0: Normal CAN operation<br>1: Restricted Operation Mode active |
| 1 | CCE | Configuration Change Enable<br>0: The CPU has no write access to the protected configuration registers.<br>1: The CPU has write access to the protected configuration registers (while MCANnCCCR.INIT = '1'). |
| 0 | INIT | Initialization<br>0: Normal Operation<br>1: Initialization is started. |

Bosch CAN FD Specification V1.0

**NOTE**

Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

### (7) MCANnBTP — Bit Timing & Prescaler Register

This register is only writable if bits MCANnCCCR.CCE and MCANnCCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 81 time quanta. The CAN time quantum may be programmed in the range of 1 to 1024 m_can_cclk (CLKP_H2) periods. tq = (BRP + 1) m_can_cclk (CLKP_H2) period.

TSEG1 is the sum of Prop_Seg and Phase_Seg1. TSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [TSEG1 + TSEG2 + 3] tq
or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $01C_H$

**Value after reset:** 0000 $0A33_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | BRP[9:0] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TSEG1[5:0] | | | | | | TSEG2[3:0] | | | | SJW[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

Bosch CAN-FD Specification V1.0

**Table 37.20 MCANnBTP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 25 to 16 | BRP[9:0] | Baud Rate Prescaler<br>$000_H$ to $3FF_H$<br>The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 15, 14 | Reserved | These bits are always read as 0. |
| 13 to 8 | TSEG1[5:0] | Time segment before sample point<br>$01_H$ to $3F_H$<br>Valid values are 1 to 63. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 7 to 4 | TSEG2[3:0] | Time segment after sample point<br>$0_H$ to $F_H$<br>Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 3 to 0 | SJW[3:0] | (Re) Synchronization Jump Width<br>$0_H$ to $F_H$<br>Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |

**NOTE**

With a CAN clock (m_can_cclk: CLKP_H2) of 8 MHz, the reset value of 0000 $0A33_H$ configures the M_CAN for a bit rate of 500 kBit/s.

### (8) MCANnTSCC — Timestamp Counter Configuration

For a description of the Timestamp Counter see **Section 37.5.3.2, Timestamp Generation**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $020_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TCP[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TSS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP |

**Table 37.21 MCANnTSCC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 19 to 16 | TCP[3:0] | Timestamp Counter Prescaler<br>$0_H$ to $F_H$<br>Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 to 16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.<br>**NOTE**<br>With CAN FD an external counter is required for timestamp generation (TSS = “10”) |
| 15 to 2 | Reserved | These bits are always read as 0. |
| 1, 0 | TSS[1:0] | Timestamp Select<br>00: Timestamp counter value always $0000_H$<br>01: Timestamp counter value incremented according to TCP<br>10: External timestamp counter value used<br>11: Same as “00” |

Bosch CAN FD Specification V1.0

### (9) MCANnTSCV — Timestamp Counter Value

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $024_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSC[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

**Table 37.22 MCANnTSCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 0 | TSC[15:0] | Timestamp Counter<br>The internal/external Timestamp Counter value is captured on start of frame (both Rx and Tx). When MCANnTSCC.TSS = “01”, the Timestamp Counter is incremented in multiples of CAN bit times [1 to 16] depending on the configuration of MCANnTSCC.TCP. A wrap around sets interrupt flag MCANnIR.TSW. Write access resets the counter to zero. When MCANnTSCC.TSS = “10”, TSC reflects the external Timestamp Counter value. A write access has no impact. |

**NOTE**

A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by write access to MCANnTSCV.

Closed (RFD Specification 0

### (10) MCANnTOCC — Timeout Counter Configuration

For a description of the Timeout Counter see **Section 37.5.3.3, Timeout Counter**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $028_H$

**Value after reset:** FFFF $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TOP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TOS[1:0] | | ETOC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP |

**Table 37.23 MCANnTOCC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TOP[15:0] | Timeout Period<br>Start value of the Timeout Counter (down-counter). Configures the Timeout Period. |
| 15 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2, 1 | TOS[1:0] | Timeout Select<br>When operating in Continuous mode, a write to MCANnTOCV presets the counter to the value configured by MCANnTOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by MCANnTOCC.TOP. Down-counting is started when the first FIFO element is stored.<br>00: Continuous operation<br>01: Timeout controlled by Tx Event FIFO<br>10: Timeout controlled by Rx FIFO 0<br>11: Timeout controlled by Rx FIFO 1 |
| 0 | ETOC | Enable Timeout Counter<br>0: Timeout Counter disabled<br>1: Timeout Counter enabled |

**NOTE**

For use of timeout function with CAN FD see **Section 37.6.3.3, Timeout Counter**.

Bosch CAN-FD Specification V1.0

### (11) MCANnTOCV — Timeout Counter Value

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $02C_H$

**Value after reset:** $0000\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TOC[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

**Table 37.24 MCANnTOCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 0 | TOC[15:0] | Timeout Counter<br>The Timeout Counter is decremented in multiples of CAN bit times [1 to 16] depending on the configuration of MCANnTSCC.TCP. When decremented to zero, interrupt flag MCANnIR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via MCANnTOCC.TOS. |

Bosch CAN FD Specification V1.0

### (12) MCANnECR — Error Counter Register

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $040_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CEL[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | X | X | X | X | X | X | X | X |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RP | REC[6:0] | | | | | | | TEC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.25 MCANnECR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. |
| 23 to 16 | CEL[7:0] | CAN Error Logging<br>The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at $FF_H$; the next increment of TEC or REC sets interrupt flag MCANnIR.ELO. |
| 15 | RP | Receive Error Passive<br>0: The Receive Error Counter is below the error passive level of 128<br>1: The Receive Error Counter has reached the error passive level of 128 |
| 14 to 8 | REC[6:0] | Receive Error Counter<br>Actual state of the Receive Error Counter, values between 0 and 127 |
| 7 to 0 | TEC[7:0] | Transmit Error Counter<br>Actual state of the Transmit Error Counter, values between 0 and 255 |

**NOTE**

When MCANnCCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

Boot CANFD Specification V1.0

### (13) MCANnPSR — Protocol Status Register

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $044_H$

**Value after reset:** $0000\ 0707_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | REDL | RBRS | RESI | FLEC[2:0] | | | BO | EW | EP | ACT[1:0] | | LEC[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| R/W | R | R | X | X | X | S | S | S | R | R | R | R | R | S | S | S |

**Table 37.26 MCANnPSR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | These bits are always read as 0. |
| 13 | REDL | Received a CAN FD Message<br>This bit is set independent of acceptance filtering.<br>0: Since this bit was reset by the CPU, no CAN FD message has been received<br>1: Message in CAN FD format with EDL flag set has been received |
| 12 | RBRS | BRS flag of last received CAN FD Message<br>This bit is set together with REDL, independent of acceptance filtering.<br>0: Last received CAN FD message did not have its BRS flag set<br>1: Last received CAN FD message had its BRS flag set |
| 11 | RESI | ESI flag of last received CAN FD Message<br>This bit is set together with REDL, independent of acceptance filtering.<br>0: Last received CAN FD message did not have its ESI flag set<br>1: Last received CAN FD message had its ESI flag set |
| 10 to 8 | FLEC[2:0] | Fast Last Error Code<br>Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. |
| 7 | BO | Bus_Off Status<br>0: The M_CAN is not Bus_Off<br>1: The M_CAN is in Bus_Off state |
| 6 | EW | Warning Status<br>0: Both error counters are below the Error_Warning limit of 96<br>1: At least one of error counter has reached the Error_Warning limit of 96 |
| 5 | EP | Error Passive<br>0: The M_CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected<br>1: The M_CAN is in the Error_Passive state |
| 4, 3 | ACT[1:0] | Activity<br>Monitors the module's CAN communication state.<br>00: Synchronizing - node is synchronizing on CAN communication<br>01: Idle - node is neither receiver nor transmitter<br>10: Receiver - node is operating as receiver<br>11: Transmitter - node is operating as transmitter |

Bosch CAN FD Specification V1.0

**Table 37.26 MCANnPSR Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 to 0 | LEC[2:0] | Last Error Code<br>The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error.<br>0: No Error: No error occurred since LEC has been reset by successful reception or transmission.<br>1: Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.<br>2: Form Error: A fixed format part of a received frame has the wrong format.<br>3: AckError: The message transmitted by the M_TTCAN was not acknowledged by another node.<br>4: Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.<br>5: Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).<br>6: CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.<br>7: NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register. |

**NOTES**

1. When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.
2. The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting MCANnCCCR.INIT. If the device goes Bus_Off, it will set MCANnCCCR.INIT of its own accord, stopping all bus activities. Once MCANnCCCR.INIT has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of MCANnCCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to MCANnPSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. MCANnECR.REC is used to count these sequences.

CPSIC CAN-FD Confidential V1.0

### (14) MCANnIR — Interrupt Register

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. The configuration of MCANnIE controls whether an interrupt is generated. The configuration of MCANnILS controls on which interrupt line an interrupt is signalled.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $050_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STE | FOE | ACKE | BE | CRCE | WDI | BO | EW | EP | ELO | BEU | BEC | DRX | TOO | MRAF | TSW |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFL | TEFF | TEFW | TEFN | TFE | TCF | TC | HPM | RF1L | RF1F | RF1W | RF1N | RF0L | RF0F | RF0W | RF0N |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.27 MCANnIR Register Contents (1/3)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | STE | Stuff Error<br>0: No Stuff Error detected<br>1: More than 5 equal bits in a sequence occurred |
| 30 | FOE | Format Error<br>0: No Format Error detected<br>1: A fixed format part of a received frame has the wrong format |
| 29 | ACKE | Acknowledge Error<br>0: No Acknowledge Error detected<br>1: A transmitted message was not acknowledged by another node |
| 28 | BE | Bit Error<br>0: No Bit Error detected<br>1: Device wanted to send a recessive / dominant level, but monitored bus level was dominant / recessive |
| 27 | CRCE | CRC Error<br>0: No CRC Error detected<br>1: Received CRC did not match the calculated CRC |
| 26 | WDI | Watchdog Interrupt<br>0: No Message RAM Watchdog event occurred<br>1: Message RAM Watchdog event due to missing READY |
| 25 | BO | Bus_Off Status<br>0: Bus_Off status unchanged<br>1: Bus_Off status changed |
| 24 | EW | Warning Status<br>0: Error_Warning status unchanged<br>1: Error_Warning status changed |
| 23 | EP | Error Passive<br>0: Error_Passive status unchanged<br>1: Error_Passive status changed |
| 22 | ELO | Error Logging Overflow<br>0: CAN Error Logging Counter did not overflow<br>1: Overflow of CAN Error Logging Counter occurred |

Bosch CAN-FD Specification V1.0

Table 37.27 MCANnIR Register Contents (2/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 21 | BEU | Bit Error Uncorrected<br>Message RAM bit error detected, uncorrected. Controlled by input signal m_can_aeim_berr[1] generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets MCANnCCCR.INIT to '1'. This is done to avoid transmission of corrupted data.<br>0: No bit error detected when reading from Message RAM<br>1: Bit error detected, uncorrected (e.g. parity logic) |
| 20 | BEC | Bit Error Corrected<br>Message RAM bit error detected and corrected. Controlled by input signal m_can_aeim_berr[0] generated by an optional external parity / ECC logic attached to the Message RAM.<br>0: No bit error detected when reading from Message RAM<br>1: Bit error detected and corrected (e.g. ECC) |
| 19 | DRX | Message stored to Dedicated Rx Buffer<br>The flag is set whenever a received message has been stored into a dedicated Rx Buffer.<br>0: No Rx Buffer updated<br>1: At least one received message stored into an Rx Buffer |
| 18 | TOO | Timeout Occurred<br>0: No timeout<br>1: Timeout reached |
| 17 | MRAF | Message RAM Access Failure<br>The flag is set, when the Rx Handler<br>• has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message.<br>• was not able to write a message to the Message RAM. In this case message storage is aborted.<br>In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location.<br>The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the M_CAN is switched into Restricted Operation Mode (see **(5) Restricted Operation Mode**). To leave Restricted Operation Mode, the Host CPU has to reset MCANnCCCR.ASM.<br>0: No Message RAM access failure occurred<br>1: Message RAM access failure occurred |
| 16 | TSW | Timestamp Wraparound<br>0: No timestamp counter wrap-around<br>1: Timestamp counter wrapped around |
| 15 | TEFL | Tx Event FIFO Element Lost<br>0: No Tx Event FIFO element lost<br>1: Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero |
| 14 | TEFF | Tx Event FIFO Full<br>0: Tx Event FIFO not full<br>1: Tx Event FIFO full |
| 13 | TEFW | Tx Event FIFO Watermark Reached<br>0: Tx Event FIFO fill level below watermark<br>1: Tx Event FIFO fill level reached watermark |
| 12 | TEFN | Tx Event FIFO New Entry<br>0: Tx Event FIFO unchanged<br>1: Tx Handler wrote Tx Event FIFO element |
| 11 | TFE | Tx FIFO Empty<br>0: Tx FIFO non-empty<br>1: Tx FIFO empty |

Bosch CAN FD Specification 2.0

Table 37.27 MCANnIR Register Contents (3/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 10 | TCF | Transmission Cancellation Finished<br>0: No transmission cancellation finished<br>1: Transmission cancellation finished |
| 9 | TC | Transmission Completed<br>0: No transmission completed<br>1: Transmission completed |
| 8 | HPM | High Priority Message<br>0: No high priority message received<br>1: High priority message received |
| 7 | RF1L | Rx FIFO 1 Message Lost<br>0: No Rx FIFO 1 message lost<br>1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero |
| 6 | RF1F | Rx FIFO 1 Full<br>0: Rx FIFO 1 not full<br>1: Rx FIFO 1 full |
| 5 | RF1W | Rx FIFO 1 Watermark Reached<br>0: Rx FIFO 1 fill level below watermark<br>1: Rx FIFO 1 fill level reached watermark |
| 4 | RF1N | Rx FIFO 1 New Message<br>0: No new message written to Rx FIFO 1<br>1: New message written to Rx FIFO 1 |
| 3 | RF0L | Rx FIFO 0 Message Lost<br>0: No Rx FIFO 0 message lost<br>1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero |
| 2 | RF0F | Rx FIFO 0 Full<br>0: Rx FIFO 0 not full<br>1: Rx FIFO 0 full |
| 1 | RF0W | Rx FIFO 0 Watermark Reached<br>0: Rx FIFO 0 fill level below watermark<br>1: Rx FIFO 0 fill level reached watermark |
| 0 | RF0N | Rx FIFO 0 New Message<br>0: No new message written to Rx FIFO 0<br>1: New message written to Rx FIFO 0 |

Bosch CAN FD Specification V1.0

### (15) MCANnIE — Interrupt Enable

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signalled on an interrupt line.

0: Interrupt disabled
1: Interrupt enabled

**Access:** This register can be read/written in 32-bit units.
**Address:** <MCANn_base> + $054_H$
**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STEE | FOEE | ACKEE | BEE | CRCEE | WDIE | BOE | EWE | EPE | ELOE | BEUE | BECE | DRXE | TOOE | MRAFE | TSWE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFLE | TEFFE | TEFWE | TEFNE | TFEE | TCFE | TCE | HPME | RF1LE | RF1FE | RF1WE | RF1NE | RF0LE | RF0FE | RF0WE | RF0NE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bosch M_CAN-FD Specification V1.0

**Table 37.28 MCANnIE Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | STEE | Stuff Error Interrupt Enable |
| 30 | FOEE | Format Error Interrupt Enable |
| 29 | ACKEE | Acknowledge Error Interrupt Enable |
| 28 | BEE | Bit Error Interrupt Enable |
| 27 | CRCEE | CRC Error Interrupt Enable |
| 26 | WDIE | Watchdog Interrupt Enable |
| 25 | BOE | Bus_Off Status Interrupt Enable |
| 24 | EWE | Warning Status Interrupt Enable |
| 23 | EPE | Error Passive Interrupt Enable |
| 22 | ELOE | Error Logging Overflow Interrupt Enable |
| 21 | BEUE | Bit Error Uncorrected Interrupt Enable |
| 20 | BECE | Bit Error Corrected Interrupt Enable |
| 19 | DRXE | Message stored to Dedicated Rx Buffer Interrupt Enable |
| 18 | TOOE | Timeout Occurred Interrupt Enable |
| 17 | MRAFE | Message RAM Access Failure Interrupt Enable |
| 16 | TSWE | Timestamp Wraparound Interrupt Enable |
| 15 | TEFLE | Tx Event FIFO Event Lost Interrupt Enable |
| 14 | TEFFE | Tx Event FIFO Full Interrupt Enable |
| 13 | TEFWE | Tx Event FIFO Watermark Reached Interrupt Enable |
| 12 | TEFNE | Tx Event FIFO New Entry Interrupt Enable |
| 11 | TFEE | Tx FIFO Empty Interrupt Enable |
| 10 | TCFE | Transmission Cancellation Finished Interrupt Enable |
| 9 | TCE | Transmission Completed Interrupt Enable |
| 8 | HPME | High Priority Message Interrupt Enable |

Table 37.28 MCANnIE Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RF1LE | Rx FIFO 1 Message Lost Interrupt Enable |
| 6 | RF1FE | Rx FIFO 1 Full Interrupt Enable |
| 5 | RF1WE | Rx FIFO 1 Watermark Reached Interrupt Enable |
| 4 | RF1NE | Rx FIFO 1 New Message Interrupt Enable |
| 3 | RF0LE | Rx FIFO 0 Message Lost Interrupt Enable |
| 2 | RF0FE | Rx FIFO 0 Full Interrupt Enable |
| 1 | RF0WE | Rx FIFO 0 Watermark Reached Interrupt Enable |
| 0 | RF0NE | Rx FIFO 0 New Message Interrupt Enable |

Bosch CAN-FD Specification V1.0

### (16) MCANnILS — Interrupt Line Select

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines.

0: Interrupt assigned to interrupt line m_can_int0 (INTMCANnI0)
1: Interrupt assigned to interrupt line m_can_int1 (INTMCANnI1)

**Access:** This register can be read/written in 32-bit units.
**Address:** <MCANn_base> + $058_H$
**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STEL | FOEL | ACKEL | BEL | CRCEL | WDIL | BOL | EWL | EPL | ELOL | BEUL | BECL | DRXL | TOOL | MRAFL | TSWL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFLL | TEFFL | TEFWL | TEFNL | TFEL | TCFL | TCL | HPML | RF1LL | RF1FL | RF1WL | RF1NL | RF0LL | RF0FL | RF0WL | RF0NL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bosch M_CAN-FD Specification V1.0

**Table 37.29 MCANnILS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | STEL | Stuff Error Interrupt Line |
| 30 | FOEL | Format Error Interrupt Line |
| 29 | ACKEL | Acknowledge Error Interrupt Line |
| 28 | BEL | Bit Error Interrupt Line |
| 27 | CRCEL | CRC Error Interrupt Line |
| 26 | WDIL | Watchdog Interrupt Line |
| 25 | BOL | Bus_Off Status Interrupt Line |
| 24 | EWL | Warning Status Interrupt Line |
| 23 | EPL | Error Passive Interrupt Line |
| 22 | ELOL | Error Logging Overflow Interrupt Line |
| 21 | BEUL | Bit Error Uncorrected Interrupt Line |
| 20 | BECL | Bit Error Corrected Interrupt Line |
| 19 | DRXL | Message stored to Dedicated Rx Buffer Interrupt Line |
| 18 | TOOL | Timeout Occurred Interrupt Line |
| 17 | MRAFL | Message RAM Access Failure Interrupt Line |
| 16 | TSWL | Timestamp Wraparound Interrupt Line |
| 15 | TEFLL | Tx Event FIFO Event Lost Interrupt Line |
| 14 | TEFFL | Tx Event FIFO Full Interrupt Line |
| 13 | TEFWL | Tx Event FIFO Watermark Reached Interrupt Line |
| 12 | TEFNL | Tx Event FIFO New Entry Interrupt Line |
| 11 | TFEL | Tx FIFO Empty Interrupt Line |
| 10 | TCFL | Transmission Cancellation Finished Interrupt Line |
| 9 | TCL | Transmission Completed Interrupt Line |
| 8 | HPML | High Priority Message Interrupt Line |

Table 37.29 MCANnILS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RF1LL | Rx FIFO 1 Message Lost Interrupt Line |
| 6 | RF1FL | Rx FIFO 1 Full Interrupt Line |
| 5 | RF1WL | Rx FIFO 1 Watermark Reached Interrupt Line |
| 4 | RF1NL | Rx FIFO 1 New Message Interrupt Line |
| 3 | RF0LL | Rx FIFO 0 Message Lost Interrupt Line |
| 2 | RF0FL | Rx FIFO 0 Full Interrupt Line |
| 1 | RF0WL | Rx FIFO 0 Watermark Reached Interrupt Line |
| 0 | RF0NL | Rx FIFO 0 New Message Interrupt Line |

Bosch CAN-FD Specification V1.0

### (17) MCANnILE — Interrupt Line Enable

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $05C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EINT1 | EINT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 37.30 MCANnILE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1 | EINT1 | Enable Interrupt Line 1<br>0: Interrupt line m_can_int1 (INTMCANnI1) disabled<br>1: Interrupt line m_can_int1 (INTMCANnI1) enabled |
| 0 | EINT0 | Enable Interrupt Line 0<br>0: Interrupt line m_can_int0 (INTMCANnI0) disabled<br>1: Interrupt line m_can_int0 (INTMCANnI0) enabled |

Bosch CAN FD Specification V1.0

### (18) MCANnGFC — Global Filter Configuration

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in **Figure 37.8, Standard Message ID Filter Path** and **Figure 37.9, Extended Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $080_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ANFS[1:0] | | ANFE[1:0] | | RRFS | RRFE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP |

**Table 37.31 MCANnGFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5, 4 | ANFS[1:0] | Accept Non-matching Frames Standard<br>Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.<br>00: Accept in Rx FIFO 0<br>01: Accept in Rx FIFO 1<br>10: Reject<br>11: Reject |
| 3, 2 | ANFE[1:0] | Accept Non-matching Frames Extended<br>Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.<br>00: Accept in Rx FIFO 0<br>01: Accept in Rx FIFO 1<br>10: Reject<br>11: Reject |
| 1 | RRFS | Reject Remote Frames Standard<br>0: Filter remote frames with 11-bit standard IDs<br>1: Reject all remote frames with 11-bit standard IDs |
| 0 | RRFE | Reject Remote Frames Extended<br>0: Filter remote frames with 29-bit extended IDs<br>1: Reject all remote frames with 29-bit extended IDs |

Bosch CAN FD Spec Rev 1.0

### (19) MCANnSIDFC — Standard ID Filter Configuration

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in **Figure 37.8, Standard Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $084_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | LSS[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLSSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.32 MCANnSIDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 16 | LSS[7:0] | List Size Standard<br>0: No standard Message ID filter<br>1 to 128: Number of standard Message ID filter elements<br>>128: Values greater than 128 are interpreted as 128 |
| 15 to 2 | FLSSA[15:2] | Filter List Standard Start Address<br>Start address of standard Message ID filter list (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

Boot CAN-ID Specification Ver.2

### (20) MCANnXIDFC — Extended ID Filter Configuration

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in **Figure 37.9, Extended Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $088_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | LSE[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLESA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.33 MCANnXIDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 22 to 16 | LSE[6:0] | List Size Extended<br>0: No extended Message ID filter<br>1 to 64: Number of extended Message ID filter elements<br>>64: Values greater than 64 are interpreted as 64 |
| 15 to 2 | FLESA[15:2] | Filter List Extended Start Address<br>Start address of extended Message ID filter list (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

Bosch CAN FD Specification V1.0

### (21) MCANnXIDAM — Extended ID AND Mask

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $090_H$

**Value after reset:** $1FFF\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | EIDM[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EIDM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

**Table 37.34 MCANnXIDAM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 28 to 0 | EIDM[28:0] | Extended ID Mask<br>For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active. |

Bosch CAN FD Specification V1.0

### (22) MCANnHPMS — High Priority Message Status

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $094_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLST | FIDX[6:0] | | | | | | | MSI[1:0] | | BIDX[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.35 MCANnHPMS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. |
| 15 | FLST | Filter List<br>Indicates the filter list of the matching filter element.<br>0: Standard Filter List<br>1: Extended Filter List |
| 14 to 8 | FIDX[6:0] | Filter Index<br>Index of matching filter element. Range is 0 to MCANnSIDFC.LSS – 1 resp. MCANnXIDFC.LSE – 1. |
| 7, 6 | MSI[1:0] | Message Storage Indicator<br>00: No FIFO selected<br>01: FIFO message lost<br>10: Message stored in FIFO 0<br>11: Message stored in FIFO 1 |
| 5 to 0 | BIDX[5:0] | Buffer Index<br>Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = '1'. |

Bosch CAN FD Specification V1.0

### (23) MCANnNDAT1 — New Data 1

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $098_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND31 | ND30 | ND29 | ND28 | ND27 | ND26 | ND25 | ND24 | ND23 | ND22 | ND21 | ND20 | ND19 | ND18 | ND17 | ND16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND15 | ND14 | ND13 | ND12 | ND11 | ND10 | ND9 | ND8 | ND7 | ND6 | ND5 | ND4 | ND3 | ND2 | ND1 | ND0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.36 MCANnNDAT1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ND[31:0] | New Data<br>The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.<br>0: Rx Buffer not updated<br>1: Rx Buffer updated from new message |

Bosch CAN FD Specification (ISO)

### (24) MCANnNDAT2 — New Data 2

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $09C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND63 | ND62 | ND61 | ND60 | ND59 | ND58 | ND57 | ND56 | ND55 | ND54 | ND53 | ND52 | ND51 | ND50 | ND49 | ND48 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND47 | ND46 | ND45 | ND44 | ND43 | ND42 | ND41 | ND40 | ND39 | ND38 | ND37 | ND36 | ND35 | ND34 | ND33 | ND32 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.37 MCANnNDAT2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ND[63:32] | New Data<br>The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.<br>0: Rx Buffer not updated<br>1: Rx Buffer updated from new message |

Bosch CAN FD Specification (ISO)

### (25) MCANnRXF0C — Rx FIFO 0 Configuration

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0A0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F0OM | F0WM[6:0] | | | | | | | — | F0S[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F0SA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.38 MCANnRXF0C Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | F0OM | FIFO 0 Operation Mode<br>FIFO 0 can be operated in blocking or in overwrite mode (see **(2) Rx FIFOs**).<br>0: FIFO 0 blocking mode<br>1: FIFO 0 overwrite mode |
| 30 to 24 | F0WM[6:0] | Rx FIFO 0 Watermark<br>0: Watermark interrupt disabled<br>1 to 64: Level for Rx FIFO 0 watermark interrupt (MCANnIR.RF0W)<br>>64: Watermark interrupt disabled |
| 23 | Reserved | This bit is always read as 0. |
| 22 to 16 | F0S[6:0] | Rx FIFO 0 Size<br>0: No Rx FIFO 0<br>1 to 64: Number of Rx FIFO 0 elements<br>>64: Values greater than 64 are interpreted as 64<br>The Rx FIFO 0 elements are indexed from 0 to F0S-1 |
| 15 to 2 | F0SA[15:2] | Rx FIFO 0 Start Address<br>Start address of Rx FIFO 0 in Message RAM (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. |

Bosch CAN FD Specification V1.0

### (26) MCANnRXF0S — Rx FIFO 0 Status

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0A4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RF0L | F0F | — | — | F0PI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | F0GI[5:0] | | | | | | — | F0FL[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.39 MCANnRXF0S Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. |
| 25 | RF0L | Rx FIFO 0 Message Lost<br>This bit is a copy of interrupt flag MCANnIR.RF0L. When MCANnIR.RF0L is reset, this bit is also reset.<br>0: No Rx FIFO 0 message lost<br>1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero<br>**NOTE**<br>Overwriting the oldest message when MCANnRXF0C.F0OM = '1' will not set this flag. |
| 24 | F0F | Rx FIFO 0 Full<br>0: Rx FIFO 0 not full<br>1: Rx FIFO 0 full |
| 23, 22 | Reserved | These bits are always read as 0. |
| 21 to 16 | F0PI[5:0] | Rx FIFO 0 Put Index<br>Rx FIFO 0 write index pointer, range 0 to 63. |
| 15, 14 | Reserved | These bits are always read as 0. |
| 13 to 8 | F0GI[5:0] | Rx FIFO 0 Get Index<br>Rx FIFO 0 read index pointer, range 0 to 63. |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 0 | F0FL[6:0] | Rx FIFO 0 Fill Level<br>Number of elements stored in Rx FIFO 0, range 0 to 64. |

Bosch CAN FD Specification V1.0

### (27) MCANnRXF0A — Rx FIFO 0 Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0A8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | F0AI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.40 MCANnRXF0A Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 to 0 | F0AI[5:0] | Rx FIFO 0 Acknowledge Index<br>After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index MCANnRXF0S.F0GI to F0AI + 1 and update the FIFO 0 Fill Level MCANnRXF0S.F0FL. |

Bosch CAN FD Specification V1.0

### (28) MCANnRXBC — Rx Buffer Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0AC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RBSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.41 MCANnRXBC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 2 | RBSA[15:2] | Rx Buffer Start Address<br>Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C. |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

Bosch CAN FD Specification V1.0

### (29) MCANnRXF1C — Rx FIFO 1 Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0B0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1OM | F1WM[6:0] | | | | | | | — | F1S[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1SA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.42 MCANnRXF1C Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | F1OM | FIFO 1 Operation Mode<br>FIFO 1 can be operated in blocking or in overwrite mode (see **(2) Rx FIFOs**).<br>0: FIFO 1 blocking mode<br>1: FIFO 1 overwrite mode |
| 30 to 24 | F1WM[6:0] | Rx FIFO 1 Watermark<br>0: Watermark interrupt disabled<br>1 to 64: Level for Rx FIFO 1 watermark interrupt (MCANnIR.RF1W)<br>>64: Watermark interrupt disabled |
| 23 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 22 to 16 | F1S[6:0] | Rx FIFO 1 Size<br>0: No Rx FIFO 1<br>1 to 64: Number of Rx FIFO 1 elements<br>>64: Values greater than 64 are interpreted as 64<br>The Rx FIFO 1 elements are indexed from 0 to F1S - 1 |
| 15 to 2 | F1SA[15:2] | Rx FIFO 1 Start Address<br>Start address of Rx FIFO 1 in Message RAM (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

Bosch CAN FD Specification V1.0

### (30) MCANnRXF1S — Rx FIFO 1 Status

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0B4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMS[1:0] | | — | — | — | — | RF1L | F1F | — | — | F1PI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | F1GI[5:0] | | | | | | — | F1FL[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.43 MCANnRXF1S Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | DMS[1:0] | Debug Message Status<br>00: Idle state, wait for reception of debug messages, DMA request is cleared<br>01: Debug message A received<br>10: Debug messages A, B received<br>11: Debug messages A, B, C received, DMA request is set |
| 29 to 26 | Reserved | These bits are always read as 0. |
| 25 | RF1L | Rx FIFO 1 Message Lost<br>This bit is a copy of interrupt flag MCANnIR.RF1L. When MCANnIR.RF1L is reset, this bit is also reset.<br>0: No Rx FIFO 1 message lost<br>1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero<br>**NOTE**<br>Overwriting the oldest message when MCANnRXF0C.F0OM = '1' will not set this flag. |
| 24 | F1F | Rx FIFO 1 Full<br>0: Rx FIFO 1 not full<br>1: Rx FIFO 1 full |
| 23, 22 | Reserved | These bits are always read as 0. |
| 21 to 16 | F1PI[5:0] | Rx FIFO 1 Put Index<br>Rx FIFO 1 write index pointer, range 0 to 63. |
| 15, 14 | Reserved | These bits are always read as 0. |
| 13 to 8 | F1GI[5:0] | Rx FIFO 1 Get Index<br>Rx FIFO 1 read index pointer, range 0 to 63. |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 0 | F1FL[6:0] | Rx FIFO 1 Fill Level<br>Number of elements stored in Rx FIFO 1, range 0 to 64. |

Bosch CAN FD Specification 1.0

### (31) MCANnRXF1A — Rx FIFO 1 Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0B8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | F1AI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.44 MCANnRXF1A Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 to 0 | F1AI[5:0] | Rx FIFO 1 Acknowledge Index<br>After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index MCANnRXF1S.F1GI to F1AI + 1 and update the FIFO 1 Fill Level MCANnRXF1S.F1FL. |

Bosch CAN FD Specification V1.0

### (32) MCANnRXESC — Rx Buffer / FIFO Element Size Configuration

Configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0BC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RBDS[2:0] | | | — | F1DS[2:0] | | | — | F0DS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | RP | RP | RP | R | RP | RP | RP | R | RP | RP | RP |

**Table 37.45 MCANnRXESC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | These bits are always read as 0. |
| 10 to 8 | RBDS[2:0] | Rx Buffer Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 4 | F1DS[2:0] | Rx FIFO 1 Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |
| 3 | Reserved | This bit is always read as 0. |
| 2 to 0 | F0DS[2:0] | Rx FIFO 0 Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |

**NOTE**

In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by MCANnRXESC are stored to the Rx Buffer resp. Rx FIFO element. The rest of the frame's data field is ignored.

Bosch M_CAN IP Specification V3.2

### (33) MCANnTXBC — Tx Buffer Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0C0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TFQM | TFQS[5:0] | | | | | | — | — | NDTB[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | RP | RP | RP | RP | RP | RP | RP | R | R | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.46 MCANnTXBC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 30 | TFQM | Tx FIFO/Queue Mode<br>0: Tx FIFO operation<br>1: Tx Queue operation |
| 29 to 24 | TFQS[5:0] | Transmit FIFO/Queue Size<br>0: No Tx FIFO/Queue<br>1 to 32: Number of Tx Buffers used for Tx FIFO/Queue<br>>32: Values greater than 32 are interpreted as 32 |
| 23, 22 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 21 to 16 | NDTB[5:0] | Number of Dedicated Transmit Buffers<br>0: No Dedicated Tx Buffers<br>1 to 32: Number of Dedicated Tx Buffers<br>>32: Values greater than 32 are interpreted as 32 |
| 15 to 2 | TBSA[15:2] | Tx Buffers Start Address<br>Start address of Tx Buffers section in Message RAM (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

**NOTE**

Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

Bosch CAN FD Specification V1.0

### (34) MCANnTXFQS — Tx FIFO/Queue Status

The Tx FIFO/Queue status is related to the pending Tx requests listed in register MCANnTXBRP. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (MCANnTXBRP not yet updated).

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0C4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | TFQF | TFQPI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | TFGI[4:0] | | | | | — | — | TFFL[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.47 MCANnTXFQS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | These bits are always read as 0. |
| 21 | TFQF | Tx FIFO/Queue Full<br>0: Tx FIFO/Queue not full<br>1: Tx FIFO/Queue full |
| 20 to 16 | TFQPI[4:0] | Tx FIFO/Queue Put Index<br>Tx FIFO/Queue write index pointer, range 0 to 31. |
| 15 to 13 | Reserved | These bits are always read as 0. |
| 12 to 8 | TFGI[4:0] | Tx FIFO Get Index<br>Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (MCANnTXBC.TFQM = '1'). |
| 7, 6 | Reserved | These bits are always read as 0. |
| 5 to 0 | TFFL[5:0] | Tx FIFO Free Level<br>Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (MCANnTXBC.TFQM = '1') |

**NOTE**

In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers.
Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

Bosch CAN FD Specification V1.0

### (35) MCANnTXESC — Tx Buffer Element Size Configuration

Configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0C8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TBDS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP |

**Table 37.48 MCANnTXESC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 to 0 | TBDS[2:0] | Tx Buffer Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |

**NOTE**

In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size MCANnTXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as "$CC_H$" (padding bytes).

Bosch M_CAN FD Specification Rev.3

### (36) MCANnTXBRP — Tx Buffer Request Pending

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0CC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRP31 | TRP30 | TRP29 | TRP28 | TRP27 | TRP26 | TRP25 | TRP24 | TRP23 | TRP22 | TRP21 | TRP20 | TRP19 | TRP18 | TRP17 | TRP16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRP15 | TRP14 | TRP13 | TRP12 | TRP11 | TRP10 | TRP9 | TRP8 | TRP7 | TRP6 | TRP5 | TRP4 | TRP3 | TRP2 | TRP1 | TRP0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.49 MCANnTXBRP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TRP[31:0] | Transmission Request Pending<br>Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register MCANnTXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register MCANnTXBCR. MCANnTXBRP bits are set only for those Tx Buffers configured via MCANnTXBC. After a MCANnTXBRP bit has been set, a Tx scan (see **Section 37.5.3.5, Tx Handling**) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID).<br>A cancellation request resets the corresponding transmission request pending bit of register MCANnTXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding MCANnTXBRP bit has been reset.<br>After a cancellation has been requested, a finished cancellation is signalled via MCANnTXBCF<br>• after successful transmission together with the corresponding MCANnTXBTO bit<br>• when the transmission has not yet been started at the point of cancellation<br>• when the transmission has been aborted due to lost arbitration<br>• when an error occurred during frame transmission<br>In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding MCANnTXBCF bit is set for all unsuccessful transmissions.<br>0: No transmission request pending<br>1: Transmission request pending<br>**NOTE**<br>MCANnTXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding MCANnTXBRP bit is reset. |

Bosch GmbH C Sample Product

### (37) MCANnTXBAR — Tx Buffer Add Request

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0D0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AR31 | AR30 | AR29 | AR28 | AR27 | AR26 | AR25 | AR24 | AR23 | AR22 | AR21 | AR20 | AR19 | AR18 | AR17 | AR16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AR15 | AR14 | AR13 | AR12 | AR11 | AR10 | AR9 | AR8 | AR7 | AR6 | AR5 | AR4 | AR3 | AR2 | AR1 | AR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.50 MCANnTXBAR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | AR[31:0] | Add Request<br>Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to MCANnTXBAR. MCANnTXBAR bits are set only for those Tx Buffers configured via MCANnTXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.<br>0: No transmission request added<br>1: Transmission requested added |

**NOTE**

If an add request is applied for a Tx Buffer with pending transmission request (corresponding MCANnTXBRP bit already set), this add request is ignored.

Preliminary Specification

### (38) MCANnTXBCR — Tx Buffer Cancellation Request

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0D4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CR31 | CR30 | CR29 | CR28 | CR27 | CR26 | CR25 | CR24 | CR23 | CR22 | CR21 | CR20 | CR19 | CR18 | CR17 | CR16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CR15 | CR14 | CR13 | CR12 | CR11 | CR10 | CR9 | CR8 | CR7 | CR6 | CR5 | CR4 | CR3 | CR2 | CR1 | CR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.51 MCANnTXBCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CR[31:0] | Cancellation Request<br>Each Tx Buffer has its own Cancellation Request bit. Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to MCANnTXBCR. MCANnTXBCR bits are set only for those Tx Buffers configured via MCANnTXBC. The bits remain set until the corresponding bit of MCANnTXBRP is reset.<br>0: No cancellation pending<br>1: Cancellation pending |

Bosch CAN FD Specification Part

### (39) MCANnTXBTO — Tx Buffer Transmission Occurred

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0D8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TO31 | TO30 | TO29 | TO28 | TO27 | TO26 | TO25 | TO24 | TO23 | TO22 | TO21 | TO20 | TO19 | TO18 | TO17 | TO16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TO15 | TO14 | TO13 | TO12 | TO11 | TO10 | TO9 | TO8 | TO7 | TO6 | TO5 | TO4 | TO3 | TO2 | TO1 | TO0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.52 MCANnTXBTO Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TO[31:0] | Transmission Occurred<br>Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding MCANnTXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register MCANnTXBAR.<br>0: No transmission occurred<br>1: Transmission occurred |

Bosch CAN FD Specification (non)

### (40) MCANnTXBCF — Tx Buffer Cancellation Finished

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0DC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF31 | CF30 | CF29 | CF28 | CF27 | CF26 | CF25 | CF24 | CF23 | CF22 | CF21 | CF20 | CF19 | CF18 | CF17 | CF16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF15 | CF14 | CF13 | CF12 | CF11 | CF10 | CF9 | CF8 | CF7 | CF6 | CF5 | CF4 | CF3 | CF2 | CF1 | CF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.53 MCANnTXBCF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CF[31:0] | Cancellation Finished<br>Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding MCANnTXBRP bit is cleared after a cancellation was requested via MCANnTXBCR. In case the corresponding MCANnTXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register MCANnTXBAR.<br>0: No transmit buffer cancellation<br>1: Transmit buffer cancellation finished |

Boscl Online Specification V1.0

### (41) MCANnTXBTIE — Tx Buffer Transmission Interrupt Enable

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0E0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIE31 | TIE30 | TIE29 | TIE28 | TIE27 | TIE26 | TIE25 | TIE24 | TIE23 | TIE22 | TIE21 | TIE20 | TIE19 | TIE18 | TIE17 | TIE16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIE15 | TIE14 | TIE13 | TIE12 | TIE11 | TIE10 | TIE9 | TIE8 | TIE7 | TIE6 | TIE5 | TIE4 | TIE3 | TIE2 | TIE1 | TIE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.54 MCANnTXBTIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TIE[31:0] | Transmission Interrupt Enable<br>Each Tx Buffer has its own Transmission Interrupt Enable bit.<br>0: Transmission interrupt disabled<br>1: Transmission interrupt enable |

### (42) MCANnTXBCIE — Tx Buffer Cancellation Finished Interrupt Enable

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0E4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIE31 | CFIE30 | CFIE29 | CFIE28 | CFIE27 | CFIE26 | CFIE25 | CFIE24 | CFIE23 | CFIE22 | CFIE21 | CFIE20 | CFIE19 | CFIE18 | CFIE17 | CFIE16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIE15 | CFIE14 | CFIE13 | CFIE12 | CFIE11 | CFIE10 | CFIE9 | CFIE8 | CFIE7 | CFIE6 | CFIE5 | CFIE4 | CFIE3 | CFIE2 | CFIE1 | CFIE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.55 MCANnTXBCIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CFIE[31:0] | Cancellation Finished Interrupt Enable<br>Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit.<br>0: Cancellation finished interrupt disabled<br>1: Cancellation finished interrupt enabled |

Bosch CAN FD Specification V1.0

### (43) MCANnTXEFC — Tx Event FIFO Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0F0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | EFWM[5:0] | | | | | | — | — | EFS[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | RP | RP | RP | RP | RP | RP | R | R | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EFSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.56 MCANnTXEFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 29 to 24 | EFWM[5:0] | Event FIFO Watermark<br>0: Watermark interrupt disabled<br>1 to 32: Level for Tx Event FIFO watermark interrupt (MCANnIR.TEFW)<br>>32: Watermark interrupt disabled |
| 23, 22 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 21 to 16 | EFS[5:0] | Event FIFO Size<br>0: Tx Event FIFO disabled<br>1 to 32: Number of Tx Event FIFO elements<br>>32: Values greater than 32 are interpreted as 32<br>The Tx Event FIFO elements are indexed from 0 to EFS – 1 |
| 15 to 2 | EFSA[15:2] | Event FIFO Start Address<br>Start address of Tx Event FIFO in Message RAM (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

Bosch CAN FD Specification V1.0

### (44) MCANnTXEFS — Tx Event FIFO Status

**Access:** This register can be read in 32-bit units.

**Address:** <MCANn_base> + $0F4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TEFL | EFF | — | — | — | EFPI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | EFGI[4:0] | | | | | — | — | EFFL[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.57 MCANnTXEFS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. |
| 25 | TEFL | Tx Event FIFO Element Lost<br>This bit is a copy of interrupt flag MCANnIR.TEFL. When MCANnIR.TEFL is reset, this bit is also reset.<br>0: No Tx Event FIFO element lost<br>1: Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero. |
| 24 | EFF | Event FIFO Full<br>0: Tx Event FIFO not full<br>1: Tx Event FIFO full |
| 23 to 21 | Reserved | These bits are always read as 0. |
| 20 to 16 | EFPI[4:0] | Event FIFO Put Index<br>Tx Event FIFO write index pointer, range 0 to 31. |
| 15 to 13 | Reserved | These bits are always read as 0. |
| 12 to 8 | EFGI[4:0] | Event FIFO Get Index<br>Tx Event FIFO read index pointer, range 0 to 31. |
| 7, 6 | Reserved | These bits are always read as 0. |
| 5 to 0 | EFFL[5:0] | Event FIFO Fill Level<br>Number of elements stored in Tx Event FIFO, range 0 to 32. |

Bosch CAN FD Specification V1.0

### (45) MCANnTXEFA — Tx Event FIFO Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MCANn_base> + $0F8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | EFAI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 37.58 MCANnTXEFA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 4 to 0 | EFAI[4:0] | Event FIFO Acknowledge Index<br>After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index MCANnTXEFS.EFGI to EFAI + 1 and update the FIFO 0 Fill Level MCANnTXEFS.EFFL. |

Bosch CAN FD Specification V1.0

### 37.5.2.4 Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration a single- or dual-ported Message RAM has to be connected to the M_CAN module.

#### (1) Message RAM Configuration

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_CAN module can be configured to allocate up to 4352 words in the Message RAM. It is not necessary to configure each of the sections listed in **Figure 37.4, Message RAM Configuration**, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via MCANnRXESC.F0DS, MCANnRXESC.F1DS, MCANnRXESC.RBDS, and MCANnTXESC.TBDS.

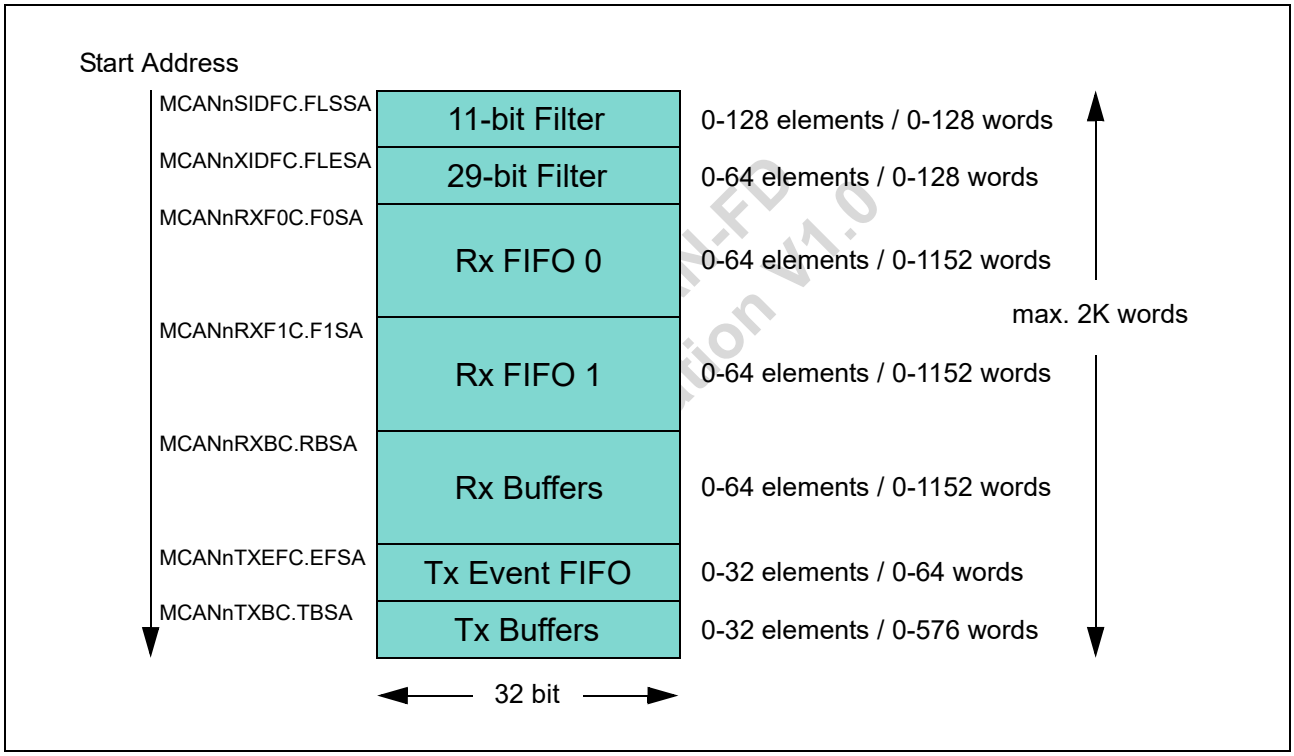



**Figure 37.4 Message RAM Configuration**

When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

**NOTE**

The M_CAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

### (2) Rx Buffer and FIFO Element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in **Table 37.59** below. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register MCANnRXESC.

**Table 37.59 Rx Buffer and FIFO Element**

| | 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|---|
| R0 | ESI, XTD, RTR, ID[28:0] | | | |
| R1 | ANMF, FIDX[6:0] | res, EDL, BRS, DLC[3:0] | RXTS[15:0] | |
| R2 | DB3[7:0] | DB2[7:0] | DB1[7:0] | DB0[7:0] |
| R3 | DB7[7:0] | DB6[7:0] | DB5[7:0] | DB4[7:0] |
| ... | ... | ... | ... | ... |
| Rn | DBm[7:0] | DBm-1[7:0] | DBm-2[7:0] | DBm-3[7:0] |

Bosch CAN-FD Specification V1.0

**R0 Bit 31 ESI: Error State Indicator**

0: Transmitting node is error active
1: Transmitting node is error passive

**R0 Bit 30 XTD: Extended Identifier**

Signals to the Host whether the received frame has a standard or extended identifier.

0: 11-bit standard identifier
1: 29-bit extended identifier

**R0 Bit 29 RTR: Remote Transmission Request**

Signals to the Host whether the received frame is a data frame or a remote frame.

0:Received frame is a data frame

1:Received frame is a remote frame

**NOTE**

There are no remote frames in CAN FD format. In case a CAN FD frame was received (EDL = 1'), bit RTR reflects the state of the reserved bit r1.

**R0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

**R1 Bit 31 ANMF: Accepted Non-matching Frame**

Acceptance of non-matching frames may be enabled via MCANnGFC.ANFS and MCANnGFC.ANFE.

0:Received frame matching filter index FIDX

1:Received frame did not match any Rx filter element

**R1 Bits 30:24 FIDX[6:0]:Filter Index**

0 to 127:Index of matching Rx acceptance filter element (invalid if ANMF = '1').
Range is 0 to MCANnSIDFC.LSS - 1 resp. MCANnXIDFC.LSE - 1.

**R1 Bit 21 EDL: Extended Data Length**

0:Standard frame format

1:CAN FD frame format (new DLC-coding and CRC)

**R1 Bit 20 BRS: Bit Rate Switch**

0:Frame received without bit rate switching

1:Frame received with bit rate switching

**R1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: received frame has 0 to 8 data bytes

9 to 15: CAN: received frame has 8 data bytes

9 to 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes

**R1 Bits 15:0 RXTS[15:0]:Rx Timestamp**

Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler MCANnTSCC.TCP.

Bosch CAN-FD Specification V1.0

**R2 Bits 31:24 DB3[7:0]: Data Byte 3**

**R2 Bits 23:16 DB2[7:0]: Data Byte 2**

**R2 Bits 15:8 DB1[7:0]: Data Byte 1**

**R2 Bits 7:0 DB0[7:0]: Data Byte 0**

**R3 Bits 31:24 DB7[7:0]: Data Byte 7**

**R3 Bits 23:16 DB6[7:0]: Data Byte 6**

**R3 Bits 15:8 DB5[7:0]: Data Byte 5**

**R3 Bits 7:0 DB4[7:0]: Data Byte 4**

**...**

**Rn Bits 31:24 DBm[7:0]: Data Byte m**

**Rn Bits 23:16 DBm-1[7:0]:Data Byte m-1**

**Rn Bits 15:8 DBm-2[7:0]:Data Byte m-2**

**Rn Bits 7:0 DBm-3[7:0]:Data Byte m-3**

Bosch CAN-FD Specification V1.0

**NOTE**

Depending on the configuration of the element size (MCANnRXESC), between two and sixteen 32-bit words (Rn = 3 to 17) are used for storage of a CAN message's data field.

### (3) Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration MCANnTXBC.TFQS and MCANnTXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register MCANnTXESC.

**Table 37.60 Tx Buffer Element**

| | 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|---|
| T0 | res, XTD, RTR, ID[28:0] | | | |
| T1 | MM[7:0] | EFC, res, DLC[3:0] | res | |
| T2 | DB3[7:0] | DB2[7:0] | DB1[7:0] | DB0[7:0] |
| T3 | DB7[7:0] | DB6[7:0] | DB5[7:0] | DB4[7:0] |
| ... | ... | ... | ... | ... |
| Tn | DBm[7:0] | DBm-1[7:0] | DBm-2[7:0] | DBm-3[7:0] |

Bosch CAN-FD Specification V1.0

**T0 Bit 30 XTD: Extended Identifier**

0: 11-bit standard identifier

1: 29-bit extended identifier

**T0 Bit 29 RTR: Remote Transmission Request**

0: Transmit data frame

1: Transmit remote frame

**NOTE**

When RTR = 1, the M_CAN transmits a remote frame according to ISO11898-1, even if MCANnCCCR.CME enables the transmission in CAN FD format.

**T0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

**T1 Bits 31:24 MM[7:0]: Message Marker**

Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.

**T1 Bit 23 EFC: Event FIFO Control**

0: Don’t store Tx events

1: Store Tx events

**T1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: transmit frame has 0-8 data bytes

9 to 15: CAN: transmit frame has 8 data bytes

9 to 15: CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes

**T2 Bits 31:24DB3[7:0]:Data Byte 3**

**T2 Bits 23:16 DB2[7:0]: Data Byte 2**

**T2 Bits 15:8 DB1[7:0]: Data Byte 1**

**T2 Bits 7:0 DB0[7:0]: Data Byte 0**

**T3 Bits 31:24 DB7[7:0]: Data Byte 7**

**T3 Bits 23:16 DB6[7:0]: Data Byte 6**

**T3 Bits 15:8 DB5[7:0]: Data Byte 5**

**T3 Bits 7:0 DB4[7:0]: Data Byte 4**

**...**

**Tn Bits 31:24 DBm[7:0]: Data Byte m**

**Tn Bits 23:16 DBm-1[7:0]:Data Byte m-1**

**Tn Bits 15:8 DBm-2[7:0]:Data Byte m-2**

**Tn Bits 7:0 DBm-3[7:0]:Data Byte m-3**

Bosch CAN-FD Specification V1.0

**NOTE**

Depending on the configuration of the element size (MCANnTXESC), between two and sixteen 32-bit words (Tn = 3 to 17) are used for storage of a CAN message’s data field.

### (4) Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register MCANnTXEFS

**Table 37.61 Tx Event FIFO Element**

| | 31 | | | | 24 | 23 | | | | 16 | 15 | | | | 8 | 7 | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E0 | ESI | XTD | RTR | | ID[28:0] | | | | | | | | | | | | | | | |
| E1 | MM[7:0] | | | | | ET[1:0] | EDL | BRS | | DLC[3:0] | | | | | TXTS[15:0] | | | | | |

**E0 Bit 31 ESI: Error State Indicator**

0: Transmitting node is error active

1: Transmitting node is error passive

**E0 Bit 30 XTD: Extended Identifier**

0: 11-bit standard identifier

1: 29-bit extended identifier

**E0 Bit 29 RTR: Remote Transmission Request**

0: Data frame transmitted

1: Remote frame transmitted

**E0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

**E1 Bits 31:24 MM[7:0]: Message Marker**

Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.

**E1 Bit 23:22 ET[1:0]: Event Type**

00: Reserved

01: Tx event

10: Transmission in spite of cancellation (always set for transmissions in DAR mode)

11: Reserved

**E1 Bit 21 EDL: Extended Data Length**

0: Standard frame format

1: CAN FD frame format (new DLC-coding and CRC)

**E1 Bit 20 BRS: Bit Rate Switch**

0: Frame transmitted without bit rate switching

1: Frame transmitted with bit rate switching

Bosch CAN-FD Specification V1.0

**E1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: frame with 0-8 data bytes transmitted

9 to 15: CAN: frame with 8 data bytes transmitted

9 to 15: CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted

**E1 Bits 15:0 TXTS[15:0]:Tx Timestamp**

Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler MCANnTSCC.TCP.

Bosch CAN-FD Specification V1.0

### (5) Standard Message ID Filter Element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address MCANnSIDFC.FLSSA plus the index of the filter element (0 to 127).

**Table 37.62 Standard Message ID Filter Element**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | SFT[1:0] | | SFEC[2:0] | | SFID1[10:0] | | | | | | | | | | | res | | | | | | | | SFID2[10:0] | | | | | | | |

**Bits 31:30 SFT[1:0]: Standard Filter Type**

00: Range filter from SF1ID to SF2ID (SF2ID ≥ SF1ID)

01: Dual ID filter for SF1ID or SF2ID

10: Classic filter: SF1ID = filter, SF2ID = mask

11: Reserved

**Bit 29:27 SFEC[2:0]:Standard Filter Element Configuration**

All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag MCANnIR.HPM and, if enabled, an interrupt is generated. In this case register MCANnHPMS is updated with the status of the priority match.

000: Disable filter element

001: Store in Rx FIFO 0 if filter matches

010: Store in Rx FIFO 1 if filter matches

011: Reject ID if filter matches

100: Set priority if filter matches

101: Set priority and store in FIFO 0 if filter matches

110: Set priority and store in FIFO 1 if filter matches

111: Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored

**Bits 26:16 SFID1[10:0]:Standard Filter ID 1**

First ID of standard ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.

Bosch CAN FD Specification V1.0

**Bits 10:0 SFID2[10:0]:Standard Filter ID 2**

This bit field has a different meaning depending on the configuration of SFEC:

1) SFEC = “001” to “110”Second ID of standard ID filter element

2) SFEC = “111”Filter for Rx Buffers or for debug messages

**SFID2[10:9]**: These bits decide whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.
00: Store message into an Rx Buffer
01: Debug Message A
10: Debug Message B
11: Debug Message C

**SFID2[8:6]**: These bits are used to control the filter event pins m_can_fe[2:0] at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_can_hclk (CLK_HSB) period in case the filter matches.

**SFID2[5:0]:** These bits define the offset to the Rx Buffer Start Address MCANnRXBC.RBSA for storage of a matching message.

Bosch CAN-FD Specification V1.0

### (6) Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address MCANnXIDFC.FLESA plus two times the index of the filter element (0 to 63).

**Table 37.63 Extended Message ID Filter Element**

| | 31 | 24 | 23 | 16 | 15 | 8 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|
| F0 | EFEC[2:0] | | EFID1[28:0] | | | | | |
| F1 | EFT[1:0] | res | EFID2[28:0] | | | | | |

#### F0 Bit 31:29 EFEC[2:0]:Extended Filter Element Configuration

All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = “100”, “101”, or “110” a match sets interrupt flag MCANnIR.HPM and, if enabled, an interrupt is generated. In this case register MCANnHPMS is updated with the status of the priority match.

000: Disable filter element

001: Store in Rx FIFO 0 if filter matches

010: Store in Rx FIFO 1 if filter matches

011: Reject ID if filter matches

100: Set priority if filter matches

101: Set priority and store in FIFO 0 if filter matches

110: Set priority and store in FIFO 1 if filter matches

111: Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored

Bosch CAN-FD Specification V1.0

#### F0 Bits 28:0 EFID1[28:0]:Extended Filter ID 1

First ID of extended ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only MCANnXIDAM masking mechanism (see **Section (e), Extended Message ID Filtering**) is used.

#### F1 Bits 31:30 EFT[1:0]: Extended Filter Type

00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID)

01: Dual ID filter for EF1ID or EF2ID

10: Classic filter: EF1ID = filter, EF2ID = mask

11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), MCANnXIDAM mask not applied

**F1 Bits 28:0 EFID2[28:0]:Extended Filter ID 2**

This bit field has a different meaning depending on the configuration of EFEC:

1) EFEC: “001” to ”110”Second ID of extended ID filter element

2) EFEC: “111”Filter for Rx Buffers or for debug messages

**EFID2[10:9]**: These bits decide whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.
00: Store message into an Rx Buffer
01: Debug Message A
10: Debug Message B
11: Debug Message C

**EFID2[8:6]**: These bits are used to control the filter event pins m_can_fe[2:0] at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_can_hclk (CLK_HSB) period in case the filter matches.

**EFID2[5:0]:** These bits define the offset to the Rx Buffer Start Address MCANnRXBC.RBSA for storage of a matching message.

Bosch CAN-FD Specification V1.0

## 37.5.3 Functional Description

### 37.5.3.1 Operating Modes

#### (1) Software Initialization

Software initialization is started by setting bit MCANnCCCR.INIT, either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While MCANnCCCR.INIT is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output m_can_tx is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting MCANnCCCR.INIT does not change any configuration register. Resetting MCANnCCCR.INIT finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (≡ Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits MCANnCCCR.INIT and MCANnCCCR.CCE are set (protected write).

MCANnCCCR.CCE can only be set/reset while MCANnCCCR.INIT = '1'. MCANnCCCR.CCE is automatically reset when MCANnCCCR.INIT is reset.

The following registers are reset when MCANnCCCR.CCE is set

- MCANnHPMS - High Priority Message Status
- MCANnRXF0S - Rx FIFO 0 Status
- MCANnRXF1S - Rx FIFO 1 Status
- MCANnTXFQS - Tx FIFO/Queue Status
- MCANnTXBRP - Tx Buffer Request Pending
- MCANnTXBTO - Tx Buffer Transmission Occurred
- MCANnTXBCF - Tx Buffer Cancellation Finished
- MCANnTXEFS - Tx Event FIFO Status

The Timeout Counter value MCANnTOCV.TOC is preset to the value configured by MCANnTOCC.TOP when MCANnCCCR.CCE is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while MCANnCCCR.CCE = '1'.

The following registers are only writable while MCANnCCCR.CCE = '0'

- MCANnTXBAR - Tx Buffer Add Request
- MCANnTXBCR - Tx Buffer Cancellation Request

MCANnCCCR.TEST and MCANnCCCR.MON can only be set by the Host while MCANnCCCR.INIT = '1' and MCANnCCCR.CCE = '1'. Both bits may be reset at any time. MCANnCCCR.DAR can only be set/reset while MCANnCCCR.INIT = '1' and MCANnCCCR.CCE = '1'.

Bosch CAN FD Specification V1.0

### (2) Normal Operation

Once the M_TTCAN is initialized and MCANnCCCR.INIT is reset to *zero*, the M_CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

### (3) CAN FD Operation

There are two variants in the CAN FD frame format, first the CAN FD frame without bit rate switching where the data field of a CAN frame may be longer than 8 bytes. The second variant is the CAN FD frame where control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame.

The CAN operation mode is enabled by programming MCANnCCCR.CME. In case MCANnCCCR.CME = “01” transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With MCANnCCCR.CME = “10”/“11” transmission and reception of long and fast CAN FD frames is enabled. MCANnCCCR.CME can only be changed while MCANnCCCR.INIT and MCANnCCCR.CCE are both set.

When initialization is left (MCANnCCCR.INIT set to ‘0’), the CAN FD protocol option is inactive, it has to be requested by writing to MCANnCCCR.CMR.

A mode change requested by writing to MCANnCCCR.CMR will be executed next time the CAN protocol controller FSM reaches idle phase between CAN frames. Upon this event MCANnCCCR.CMR is reset to “00” and the status flags MCANnCCCR.FDBS and MCANnCCCR.FDO are set accordingly. In case the requested CAN operation mode is not enabled, the value written to MCANnCCCR.CMR is retained until it is overwritten by the next mode change request. Default is CAN operation according to ISO11898-1.

Bosch CAN-FD Specification V1.0

It is not necessary to change the CAN operation mode after system startup. A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting according to ISO11898-1 until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in silent mode until programming has completed. Then all nodes switch back to CAN communication according ISO11898-1.

When MCANnCCCR.CME [1] “00”, received CAN FD frames are interpreted according to the CAN FD Protocol Specification. The reserved bit in CAN frames with 11-bit identifiers and the first reserved bit in CAN frames with 29-bit identifiers will be decoded as EDL bit. EDL = recessive signifies a CAN FD frame, EDL = dominant signifies a standard CAN frame. In a CAN FD frame, the two bits following EDL, r0 and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by r0 = dominant and BRS = recessive. The coding of r0 = recessive is reserved for future expansion of the protocol.

Reception of CAN frames according to ISO 11898-1 is possible in all CAN operation modes.

The status bits MCANnCCCR.FDO and MCANnCCCR.FDBS indicate the format of transmitted frames. When MCANnCCCR.FDO is set, frames will be transmitted in CAN FD format with EDL = recessive. When both MCANnCCCR.FDO and MCANnCCCR.FDBS are set, frames will be transmitted in CAN FD format with bit rate switching and both bits EDL and BRS = recessive.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to **Table 37.64, Coding of DLC in CAN FD** below.

**Table 37.64 Coding of DLC in CAN FD**

| DLC | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| Number of Data Bytes | 12 | 16 | 20 | 24 | 32 | 48 | 64 |

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing & Prescaler Register MCANnBTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the Fast Bit Timing & Prescaler Register MCANnFBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (m_can_cclk: CLKP_H2). Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

Bosch CAN FD Specification V1.0

### (4) Transmitter Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin m_can_tx the protocol controller receives the transmitted data from its local CAN transceiver via pin m_can_rx. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transceiver loop delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

#### (a) Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver. **Figure 37.5, Transmitter delay measurement** below describes how the transceiver loop delay is measured.

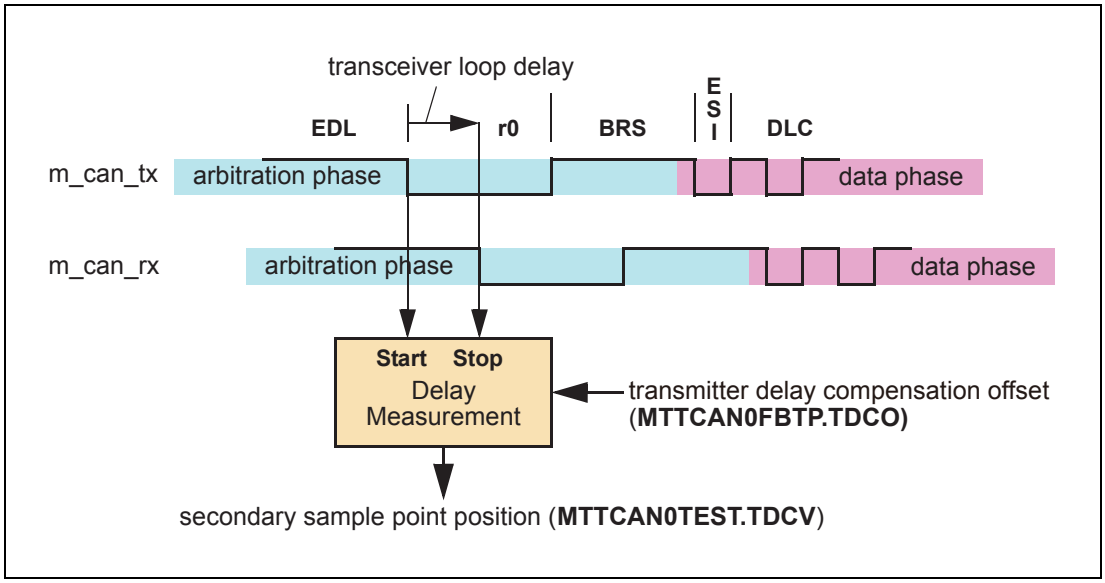



**Figure 37.5 Transmitter delay measurement**

Within each CAN FD frame, the transmitter measures the delay between the data transmitted at pin m_can_tx and the data received at pin m_can_rx. The measurement is done once, at the falling edge of bit EDL to bit r0. The delay is measured in m_can_cclk (CLKP_H2) periods.

A secondary sample point position is calculated by adding a configurable transmitter delay compensation offset MCANnFBTP.TDCO to the measured transceiver delay. This transmitter delay compensation value MCANnTEST.TDCV is the sum of the measured transceiver delay and the transmitter delay compensation offset. The transmitter delay compensation offset is chosen to adjust the secondary sample point inside the bit time (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of time quanta tq.

To check for bit errors during the data phase, the delayed transmit data is compared against the received data at the secondary sample point. If a bit error is detected at the secondary sample point, the transmitter will react to this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

For the transmitter delay compensation the following boundary conditions have to be considered:

- The sum of the measured delay from m_can_tx to m_can_rx and the configured transmitter delay compensation offset MCANnFBTP.TDCO has to be less than 3 bit times in the data phase.
- The sum of the measured delay from m_can_tx to m_can_rx and the configured transmitter delay compensation offset MCANnFBTP.TDCO has to be less or equal 63 m_can_cclk (CLKP_H2) periods. In case this sum exceeds 63 m_can_cclk (CLKP_H2) periods, the maximum value of 63 m_can_cclk (CLKP_H2) periods is used for transmitter delay compensation.

The actual delay compensation value is monitored by reading MCANnTEST.TDCV.

(b) Configuration and Status

Compensation for the transceiver loop delay by the M_CAN is enabled via MCANnFBTP.TDC. The transmitter delay compensation offset is configured via MCANnFBTP.TDCO. The actual delay compensation value applied by the M_CAN's protocol engine can be read from MCANnTEST.TDCV.

Bosch CAN FD Specification V1.0

### (5) Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. The Host can set the M_CAN into Restricted Operation mode by setting bit MCANnCCCR.ASM. The bit can only be set by the Host when both MCANnCCCR.CCE and MCANnCCCR.INIT are set to '1'. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset MCANnCCCR.ASM.

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

If the M_CAN is connected to a Clock Calibration on CAN unit, MCANnCCCR.ASM is controlled by input m_can_cok. In case m_can_cok switches to '0', bit MCANnCCCR.ASM is set. When m_can_cok switches back to '1', bit MCANnCCCR.ASM returns to the previously written value. The state of MCANnCCCR.ASM is the written value while input m_can_cok is at '1'. The input is hardwired to '1' when there is no Clock Calibration on CAN unit connected.

Bosch CAN FD Specification V1.0

### (6) Bus Monitoring Mode

The M_CAN is set in Bus Monitoring Mode by programming MCANnCCCR.MON to one. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the M_TTCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the M_TTCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_TTCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode register MCANnTXBRP is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. **Figure 37.6** shows the connection of signals m_can_tx and m_can_rx to the M_CAN in Bus Monitoring Mode.

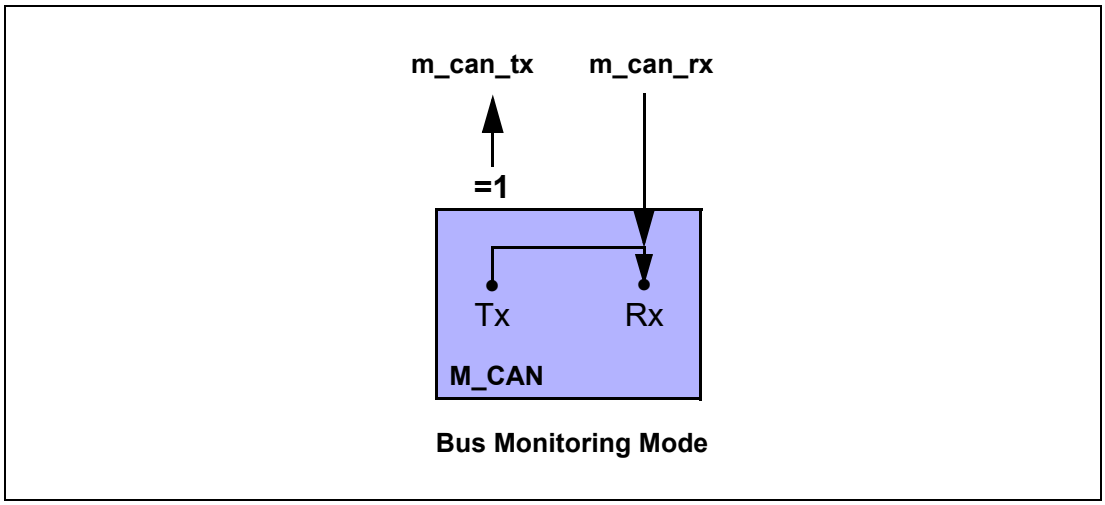



Figure 37.6 Pin Control in Bus Monitoring Mode

### (7) Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the M_TTCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via MCANnCCCR.DAR.

#### (a) Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit MCANnTXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  Corresponding Tx Buffer Transmission Occurred bit MCANnTXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit MCANnTXBCF.CFx not set
- Successful transmission in spite of cancellation:
  Corresponding Tx Buffer Transmission Occurred bit MCANnTXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit MCANnTXBCF.CFx set
- Arbitration lost or frame transmission disturbed:
  Corresponding Tx Buffer Transmission Occurred bit MCANnTXBTO.TOx not set
  Corresponding Tx Buffer Cancellation Finished bit MCANnTXBCF.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = "10" (transmission in spite of cancellation).

Bosch CAN FD Specification V1.0

### (8) Power Down (Sleep Mode)

The M_TTCAN can be set into power down mode by using CC Control Register MCANnCCCR.CSR. When all pending transmission requests have completed, the M_TTCAN waits until bus idle state is detected. Then the M_TTCAN sets then MCANnCCCR.INIT to one to prevent any further CAN transfers. Now the M_TTCAN acknowledges that it is ready for power down by setting MCANnCCCR.CSA to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to MCANnCCCR.INIT will have no effect. Now the module clock inputs m_can_hclk (CLK_HSB) and m_can_cclk (CLKP_H2) may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting CC Control Register flag MCANnCCCR.CSR. The M_CAN will acknowledge this by resetting MCANnCCCR.CSA. Afterwards, the application can restart CAN communication by resetting bit MCANnCCCR.INIT.

### (9) Test Modes

To enable write access to register MCANnTEST (see **(4) MCANnTEST — Test Register**), bit MCANnCCCR.TEST has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin m_can_tx by programming MCANnTEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_CAN's bit timing and it can drive constant dominant or recessive values. The actual value at pin m_can_rx can be read from MCANnTEST.RX. Both functions can be used to check the CAN bus' physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to MCANnTEST.TX until the new configuration is visible at output pin m_can_tx. This applies also when reading input pin m_can_rx via MCANnTEST.RX.

**NOTE**

Test modes should be used for production tests or self test only. The software control for pin m_can_tx interferes with all CAN protocol functions. It is not recommended to use test modes for application.

(a) External Loop Back Mode

The M_CAN can be set in External Loop Back Mode by programming MCANnTEST.LBCK to one. In Loop Back Mode, the M_CAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx Buffer or an Rx FIFO. **Figure 37.7, Pin Control in Loop Back Modes** shows the connection of signals m_can_tx and m_can_rx to the M_CAN in External Loop Back Mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the M_CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the m_can_rx input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the m_can_tx pin.

(b) Internal Loop Back Mode

Internal Loop Back Mode is entered by programming bits MCANnTEST.LBCK and MCANnCCCR.MON to one. This mode can be used for a “Hot Selftest”, meaning the M_TTCAN can be tested without affecting a running CAN system connected to the pins m_can_tx and m_can_rx. In this mode pin m_can_rx is disconnected from the M_CAN and pin m_can_tx is held recessive. **Figure 37.7, Pin Control in Loop Back Modes** shows the connection of m_can_tx and m_can_rx to the M_CAN in case of Internal Loop Back Mode.

Bosch CAN-FD Specification V1.0

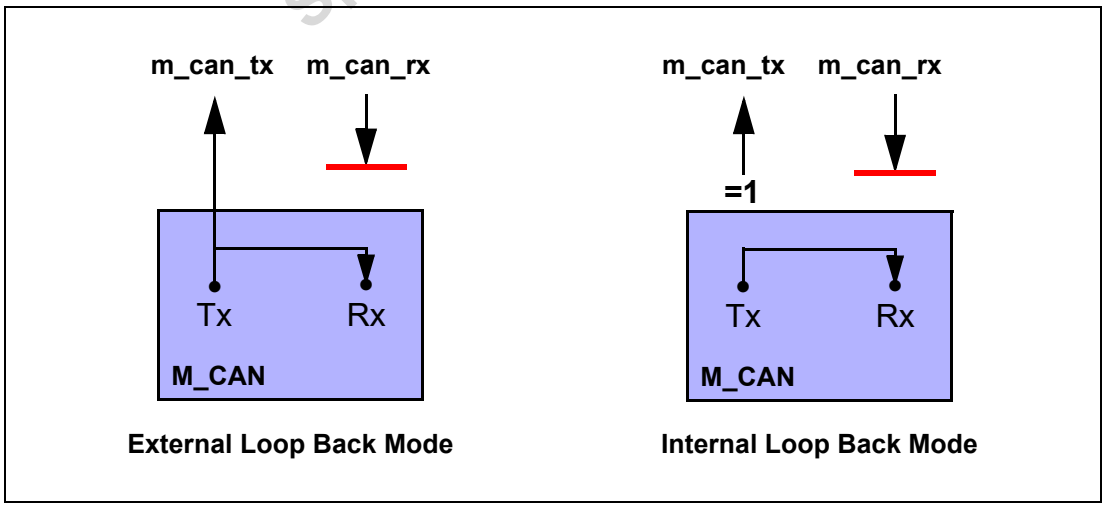



Figure 37.7 Pin Control in Loop Back Modes

#### 37.5.3.2 Timestamp Generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler MCANnTSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via MCANnTSCV.TSC. A write access to register MCANnTSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag MCANnIR.TSW is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer / Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit MCANnTSCC.TSS an external 16-bit timestamp can be used.

#### 37.5.3.3 Timeout Counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_CAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by MCANnTSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via register MCANnTOCC. The actual counter value can be read from MCANnTOCV.TOC. The Timeout Counter can only be started while MCANnCCCR.INIT = '0'. It is stopped when MCANnCCCR.INIT = '1', e.g. when the M_CAN enters Bus_Off state.

The operation mode is selected by MCANnTOCC.TOS. When operating in Continuous Mode, the counter starts when MCANnCCCR.INIT is reset. A write to MCANnTOCV presets the counter to the value configured by MCANnTOCC.TOP and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by MCANnTOCC.TOP. Down-counting is started when the first FIFO element is stored. Writing to MCANnTOCV has no effect.

When the counter reaches zero, interrupt flag MCANnIR.TOO is set. In Continuous Mode, the counter is immediately restarted at MCANnTOCC.TOP.

**NOTE**

The clock signal for the Timeout Counter is derived from the CAN Core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

Bosch CAN-FD Specification V1.0

### 37.5.3.4 Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

#### (1) Acceptance Filtering

The M_CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration MCANnGFC
- Standard ID Filter Configuration MCANnSIDFC
- Extended ID Filter Configuration MCANnXIDFC
- Extended ID AND Mask MCANnXIDAM

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag MCANnIR.HPM
- Set High Priority Message interrupt flag MCANnIR.HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

Bosch CAN FD Specification V1.0

**Rx Buffer**

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see MCANnPSR.LEC respectively MCANnPSR.FLEC.

**Rx FIFO**

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see MCANnPSR.LEC respectively MCANnPSR.FLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in **(b) Rx FIFO Overwrite Mode** have to be considered.

**NOTE**

When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

(a) Range Filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

EFT: “00”: The Message ID of received frames is ANDed with the Extended ID AND Mask (MCANnXIDAM) before the range filter is applied

EFT: “11”: The Extended ID AND Mask (MCANnXIDAM) is not used for range filtering

(b) Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

(c) Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

Bosch CAN-FD Specification V1.0

(d) Standard Message ID Filtering

**Figure 37.8, Standard Message ID Filter Path** below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in **(5) Standard Message ID Filter Element**.

Controlled by the Global Filter Configuration MCANnGFC and the Standard ID Filter Configuration MCANnSIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

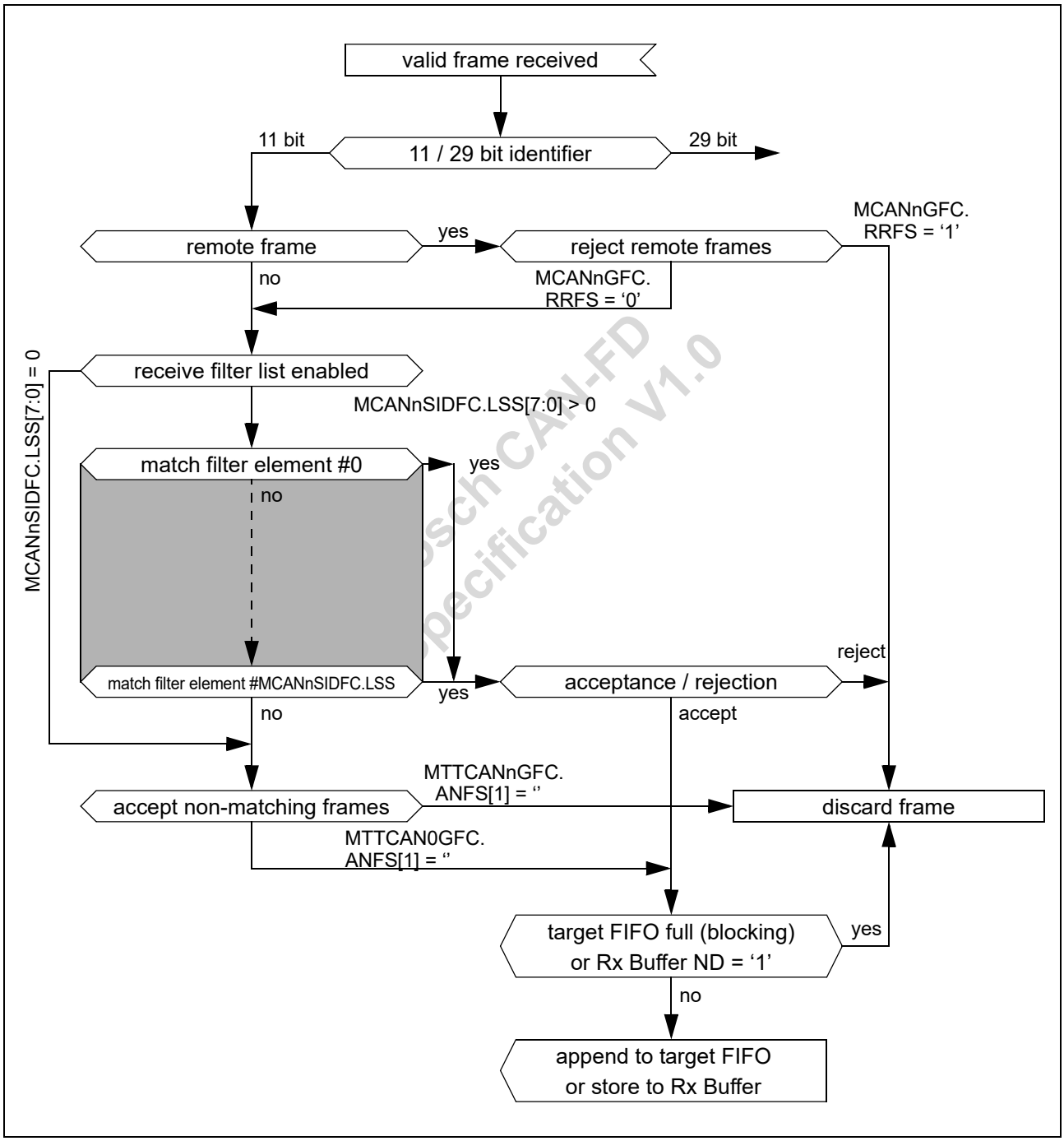



Figure 37.8 Standard Message ID Filter Path

(e) Extended Message ID Filtering

**Figure 37.9, Extended Message ID Filter Path** below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in **(6) Extended Message ID Filter Element**.

Controlled by the Global Filter Configuration MCANnGFC and the Extended ID Filter Configuration MCANnXIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask MCANnXIDAM is ANDed with the received identifier before the filter list is executed.

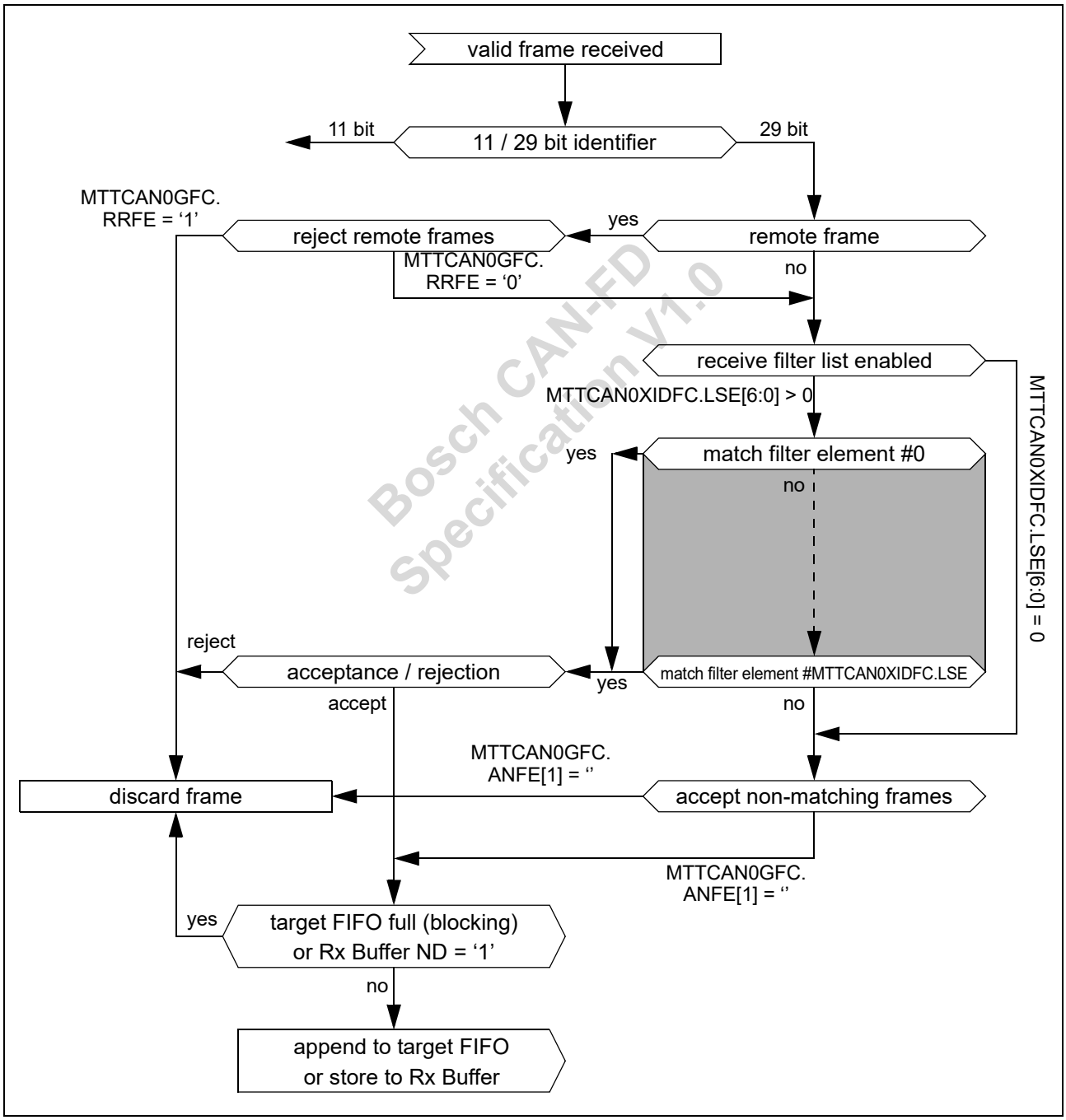



**Figure 37.9 Extended Message ID Filter Path**

### (2) Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers MCANnRXF0C and MCANnRXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see **(1) Acceptance Filtering**. The Rx FIFO element is described in **(2) Rx Buffer and FIFO Element**.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag MCANnIR.RFnW is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS.FnF. In addition interrupt flag MCANnIR.RFnF is set.

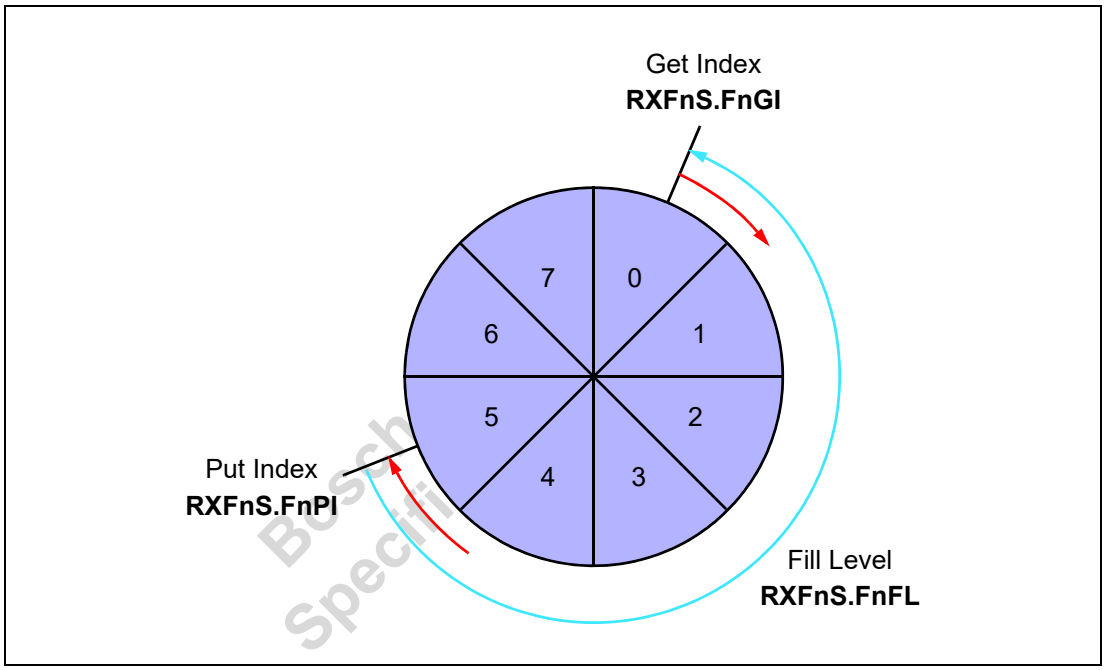



**Figure 37.10 Rx FIFO Status**

When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI • FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

**Table 37.65 Rx Buffer / FIFO Element Size**

| MCANnRXESC.RBDS[2:0]<br>MCANnRXESC.FnDS[2:0] | Data Field<br>[bytes] | FIFO Element Size<br>[RAM words] |
|---|---|---|
| 000 | 8 | 4 |
| 001 | 12 | 5 |
| 010 | 16 | 6 |
| 011 | 20 | 7 |
| 100 | 24 | 8 |
| 101 | 32 | 10 |
| 110 | 48 | 14 |
| 111 | 64 | 18 |

(a) Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = '0'. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS.FnF = '1'. In addition interrupt flag MCANnIR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = '1'. In addition interrupt flag MCANnIR.RFnL is set.

(b) Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = '1'.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signalled by RXFnS.FnF = '1', the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. **Figure 37.11, Rx FIFO Overflow Handling** shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

Bosch CAN FD Specification V1.0

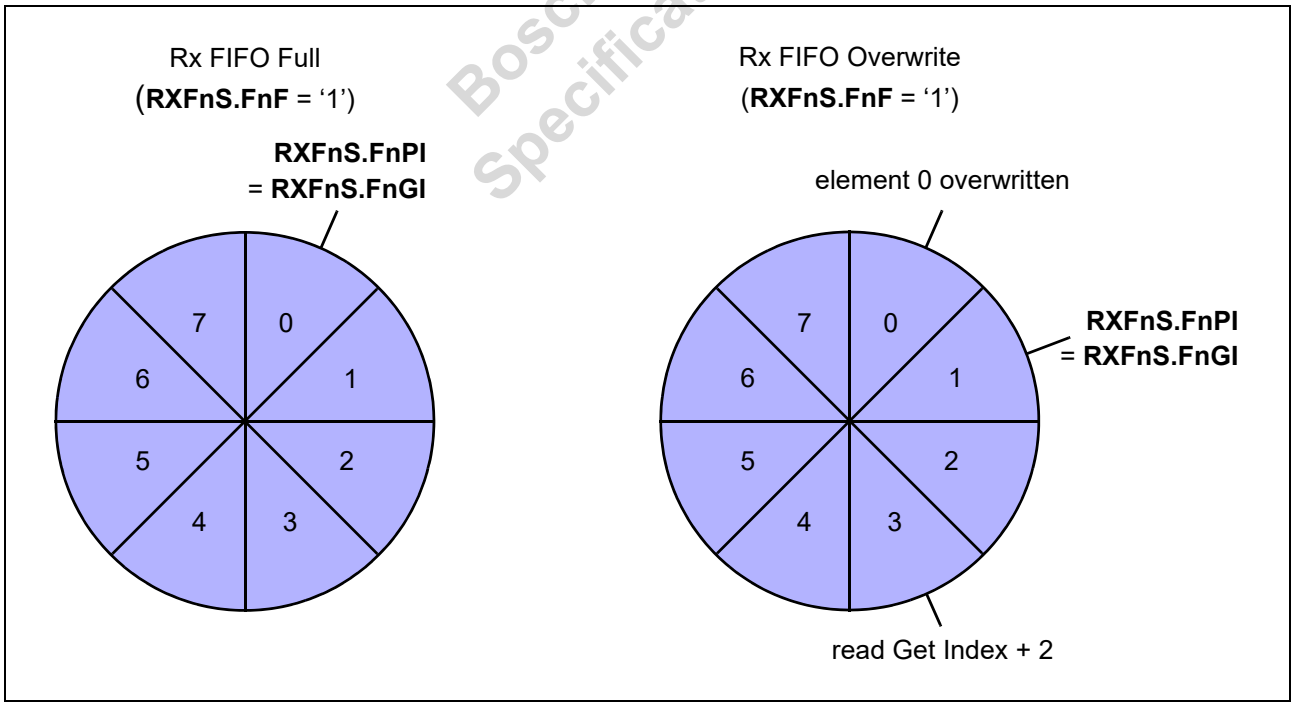



**Figure 37.11 Rx FIFO Overflow Handling**

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = '0').

### (3) Dedicated Rx Buffers

The M_CAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via MCANnRXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = “111” and SFID2 / EFID2[10:9] = “00” has to be configured (see **(5) Standard Message ID Filter Element** and **(6) Extended Message ID Filter Element**).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag MCANnIR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

**Table 37.66 Example Filter Configuration for Rx Buffers**

| Filter Element | SFID1[10:0] EFID1[28:0] | SFID2[10:9] EFID2[10:9] | SFID2[5:0] EFID2[5:0] |
|---|---|---|---|
| 0 | ID message 1 | 00 | 00 0000 |
| 1 | ID message 2 | 00 | 00 0001 |
| 2 | ID message 3 | 00 | 00 0010 |

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register MCANnNDAT1, 2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a ‘1’ to the respective bit position.

While an Rx Buffer’s New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

Bosch CAN FD Specification V1.0

#### (a) Rx Buffer Handling

- Reset interrupt flag MCANnIR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

### 37.5.3.5 Tx Handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in **(3) Tx Buffer Element**.

**NOTE**

AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register MCANnTXBRP is updated, or when a transmission has been started.

#### (1) Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

Bosch CAN-FD Specification V1.0

The transmit pause feature is controlled by bit MCANnCCCR.TXP. If the bit is set, the M_CAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (MCANnCCCR.TXP = '0').

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

#### (2) Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via MCANnTXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (see **Table 37.67, Tx Buffer / FIFO / Queue Element Size**). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0 to 31) • Element Size to the Tx Buffer Start Address MCANnTXBC.TBSA.

Table 37.67 Tx Buffer / FIFO / Queue Element Size

| MCANnTXESC.TBDS[2:0] | Data Field [bytes] | Element Size [RAM words] |
|---|---|---|
| 000 | 8 | 4 |
| 001 | 12 | 5 |
| 010 | 16 | 6 |
| 011 | 20 | 7 |
| 100 | 24 | 8 |
| 101 | 32 | 10 |
| 110 | 48 | 14 |
| 111 | 64 | 18 |

### (3) Tx FIFO

Tx FIFO operation is configured by programming MCANnTXBC.TFQM to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index MCANnTXFQS.TFGI. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_CAN calculates the Tx FIFO Free Level MCANnTXFQS.TFFL as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index MCANnTXFQS.TFQPI. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (MCANnTXFQS.TFQF = '1') is signalled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the MCANnTXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via MCANnTXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see **Table 37.67, Tx Buffer / FIFO / Queue Element Size**). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index MCANnTXFQS.TFQPI (0 to 31) • Element Size to the Tx Buffer Start Address MCANnTXBC.TBSA.

Bosch CAN-FD Specification V1.0

### (4) Tx Queue

Tx Queue operation is configured by programming MCANnTXBC.TFQM to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index MCANnTXFQS.TFQPI. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (MCANnTXFQS.TFQF = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register MCANnTXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see **Table 37.67, Tx Buffer / FIFO / Queue Element Size**). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index MCANnTXFQS.TFQPI (0 to 31) • Element Size to the Tx Buffer Start Address MCANnTXBC.TBSA.

### (5) Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by MCANnTXBC.NDTB. The number of Tx Buffers assigned to the Tx FIFO is configured by MCANnTXBC.TFQS. In case MCANnTXBC.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

Bosch CAN FD Specification V1.0

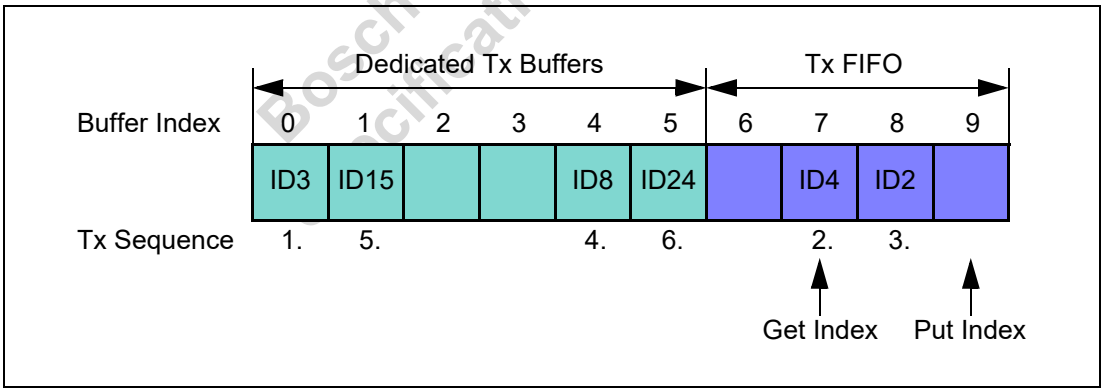



**Figure 37.12 Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO**

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS.TFGI)
- Buffer with lowest Message ID gets highest priority and is transmitted next

### (6) Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by MCANnTXBC.NDTB. The number of Tx Queue Buffers is configured by MCANnTXBC.TFQS. In case MCANnTXBC.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

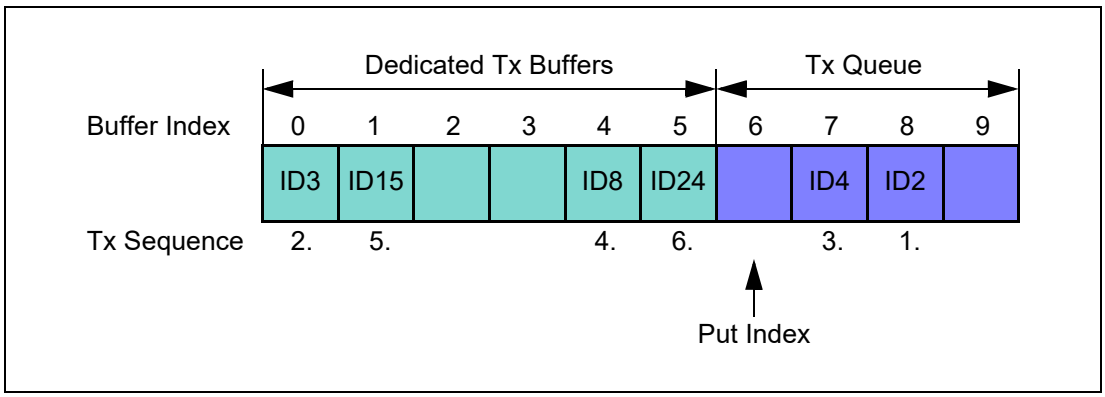



**Figure 37.13 Example of mixed Configuration Dedicated Tx Buffers / Tx Queue**

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

### (7) Transmit Cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (= number of Tx Buffer) of register MCANnTXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signalled by setting the corresponding bit of register MCANnTXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding MCANnTXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding MCANnTXBTO and MCANnTXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding MCANnTXBCF bit is set.

**NOTE**

In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

Bosch CAN FD Specification V1.0

### (8) Tx Event Handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in **(4) Tx Event FIFO Element**.

When a Tx Event FIFO full condition is signalled by MCANnIR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag MCANnIR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by MCANnTXEFC.EFWM, interrupt flag MCANnIR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index MCANnTXEFS.EFGI has to be added to the Tx Event FIFO start address MCANnTXEFC.EFSA.

Bosch CAN-FD Specification V1.0

#### 37.5.3.6 FIFO Acknowledge Handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see **(27) MCANnRXF0A — Rx FIFO 0 Acknowledge**, **(31) MCANnRXF1A — Rx FIFO 1 Acknowledge**, and **(45) MCANnTXEFA — Tx Event FIFO Acknowledge**). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus *one* and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.

When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the M_CAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

**NOTE**

The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

Bosch CAN FD Specification V1.0

# 37.6 M_TTCAN

## 37.6.1 Overview

The M_TTCAN module is the new TTCAN Communication Controller IP-module. The M_TTCAN performs communication according to ISO 11898-1 (identical to Bosch CAN protocol specification 2.0 part A,B) and according to ISO 11898-4 (Time-triggered communication on CAN). The M_TTCAN provides all features of time-triggered communication specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation. In addition the M_TTCAN supports communication according to CAN FD protocol specification 1.0. The CAN FD option can be used together with event-triggered and time-triggered CAN communication.

The message storage is intended to be a single--ported Message RAM outside of the module. It is connected to the M_TTCAN via the Generic Master Interface.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN Core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN Core as well as providing transmit status information. It implements all functions concerning the time schedule and the global system time.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_TTCAN's clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FM-PLL.

Bosch CAN-FD Specification V1.0

#### 37.6.1.1 Features

- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1, -4
- CAN FD with up to 64 data bytes supported
- TTCAN protocol level 1 and level 2 completely in hardware
- Event synchronized time-triggered communication supported
- CAN Error Logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signalling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Programmable loop-back test mode
- Maskable module interrupts
- 8/16/32 bit Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support

Bosch CAN-FD Specification V1.0

### 37.6.1.2 Block Diagram

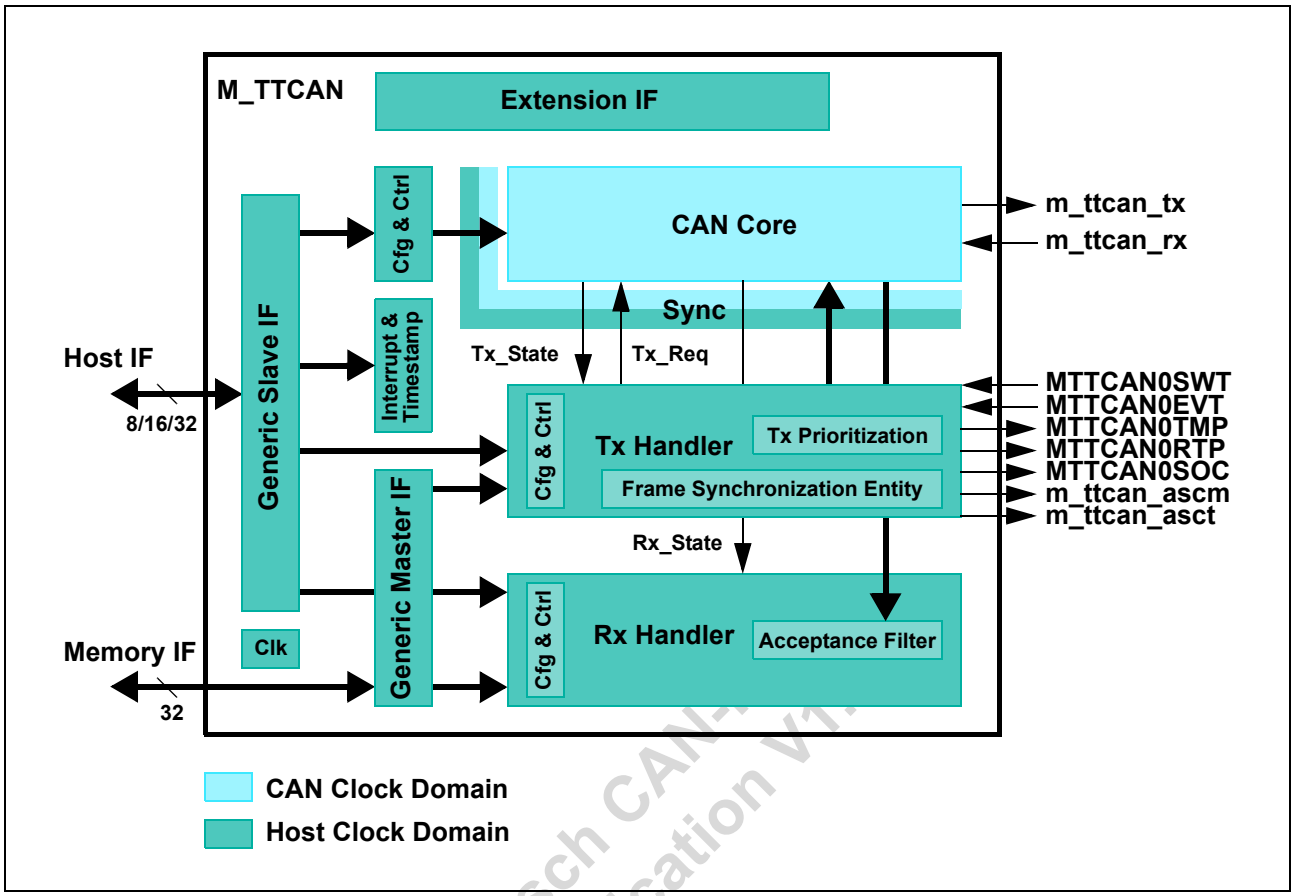



Figure 37.14 M_TTCAN Block Diagram

#### CAN Core

CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.

#### Sync

Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.

#### Clk

Synchronizes reset signal to the Host clock domain and to the CAN clock domain.

#### Cfg & Ctrl

CAN Core related configuration and control bits.

#### Interrupt & Timestamp

Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation. An externally generated 16-bit vector may substitute the integrated 16-bit CAN bit time counter for receive and transmit timestamp generation.

### Tx Handler

Controls the message transfer from the external Message RAM to the CAN Core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.

The Tx Handler also implements the Frame Synchronization Entity FSE which controls time-triggered communication according to ISO11898-4. It synchronizes itself to the reference messages on the CAN bus, controls cycle time and global time, and handles transmissions according to the predefined message schedule, the system matrix. It also handles the time marks of the system matrix that are linked to the messages in the Message RAM. Stop Watch Trigger, Event Trigger, and Time Mark Interrupt are synchronization interfaces.

### Rx Handler

Controls the transfer of received messages from the CAN Core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

Bosch CAN-FD Specification V1.0

### Generic Slave Interface

Connects the M_TTCAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8/16/32-bit bus to support a wide range of interconnection structures.

### Generic Master Interface

Connects the M_TTCAN to a local 32-bit Message RAM. The implemented Message RAM size is 2K • 32 bit.

### Extension Interface

All flags from the Interrupt Register MTTCAN0IR and TT Interrupt Register MTTCAN0TTIR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M_TTCAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

#### 37.6.1.3 Dual Clock Sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain m_ttcan_hclk (CLK_HSB). Due to the high precision clocking requirements of the CAN Core, a separate clock without any modulation has to be provided as m_ttcan_cclk (CLKP_H2). The CAN Core should be programmed to have at least 8 clocks per bittime, this is e.g. 1 Mbaud @ m_ttcan_cclk (CLKP_H2) >= 8 MHz. Even in case of a very high Host clock frequency, the clock frequency of the CAN Core needs not to be higher than 8 MHz.

Within the M_TTCAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

**NOTE**

In order to achieve a stable function of the M_TTCAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

#### 37.6.1.4 Dual Interrupt Lines

The module provides two interrupt lines. Interrupts can be routed either to m_ttcan_int0 (INTMTTCANI0) or to m_ttcan_int1(INTMTTCANI1). By default all interrupts are routed to interrupt line m_ttcan_int0 (INTMTTCANI0). By programming MTTCAN0ILE.EINT0 and MTTCAN0ILE.EINT1 the interrupt lines can be enabled or disabled separately.

Bosch CAN-FD Specification V1.0

## 37.6.2 Programmer's Model

### 37.6.2.1 Hardware Reset Description

After hardware reset, the registers of the M_TTCAN hold the reset values listed in **Table 37.68**. Additionally the Bus_Off state is reset and the output m_ttcan_tx is set to *recessive* (HIGH). The value $0001_H$ (MTTCAN0CCCR.INIT = '1') in the CC Control Register enables software initialization. The M_TTCAN does not influence the CAN bus until the CPU resets MTTCAN0CCCR.INIT to '0'.

### 37.6.2.2 Register Map

The M_TTCAN module allocates an address space of 512 bytes. All registers are organized as 32-bit registers. The M_TTCAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8 bit (byte access), 16 bit (half-word access), or 32 bit (word access). Write access by the Host CPU to registers/bits marked with "P = Protected Write" is possible only with MTTCAN0CCCR.CCE = '1' AND MTTCAN0CCCR.INIT = '1'. There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

**Table 37.68 M_TTCAN Register Map (1/2)**

| ADDRESS | SYMBOL | NAME | PAGE | RESET | ACC |
|---|---|---|---|---|---|
| <MTTCAN0_base> + $000_H$ | MTTCAN0CREL | Core Release Register | 2922 | 3013 0508 | R |
| <MTTCAN0_base> + $004_H$ | MTTCAN0ENDN | Endian Register | 2923 | 8765 4321 | R |
| <MTTCAN0_base> + $00C_H$ | MTTCAN0FBTP | Fast Bit Timing & Prescaler Register | 2924 | 0000 0A33 | RP |
| <MTTCAN0_base> + $010_H$ | MTTCAN0TEST | Test Register | 2926 | 0000 0000 | RP |
| <MTTCAN0_base> + $014_H$ | MTTCAN0RWD | RAM Watchdog | 2927 | 0000 0000 | RP |
| <MTTCAN0_base> + $018_H$ | MTTCAN0CCCR | CC Control Register | 2928 | 0000 0001 | RWPp |
| <MTTCAN0_base> + $01C_H$ | MTTCAN0BTP | Bit Timing & Prescaler Register | 2930 | 0000 0A33 | RP |
| <MTTCAN0_base> + $020_H$ | MTTCAN0TSCC | Timestamp Counter Configuration | 2931 | 0000 0000 | RP |
| <MTTCAN0_base> + $024_H$ | MTTCAN0TSCV | Timestamp Counter Value | 2932 | 0000 0000 | RC |
| <MTTCAN0_base> + $028_H$ | MTTCAN0TOCC | Timeout Counter Configuration | 2933 | FFFF 0000 | RP |
| <MTTCAN0_base> + $02C_H$ | MTTCAN0TOCV | Timeout Counter Value | 2934 | 0000 FFFF | RC |
| <MTTCAN0_base> + $030_H$ to $03C_H$ | | reserved (4) | | 0000 0000 | R |
| <MTTCAN0_base> + $040_H$ | MTTCAN0ECR | Error Counter Register | 2935 | 0000 0000 | RX |
| <MTTCAN0_base> + $044_H$ | MTTCAN0PSR | Protocol Status Register | 2936 | 0000 0707 | RXS |
| <MTTCAN0_base> + $048_H$ to $04C_H$ | | reserved (2) | | 0000 0000 | R |
| <MTTCAN0_base> + $050_H$ | MTTCAN0IR | Interrupt Register | 2938 | 0000 0000 | RW |
| <MTTCAN0_base> + $054_H$ | MTTCAN0IE | Interrupt Enable | 2941 | 0000 0000 | RW |
| <MTTCAN0_base> + $058_H$ | MTTCAN0ILS | Interrupt Line Select | 2943 | 0000 0000 | RW |
| <MTTCAN0_base> + $05C_H$ | MTTCAN0ILE | Interrupt Line Enable | 2945 | 0000 0000 | RW |
| <MTTCAN0_base> + $060_H$ to $07C_H$ | | reserved (8) | | 0000 0000 | R |
| <MTTCAN0_base> + $080_H$ | MTTCAN0GFC | Global Filter Configuration | 2946 | 0000 0000 | RP |
| <MTTCAN0_base> + $084_H$ | MTTCAN0SIDFC | Standard ID Filter Configuration | 2947 | 0000 0000 | RP |
| <MTTCAN0_base> + $088_H$ | MTTCAN0XIDFC | Extended ID Filter Configuration | 2948 | 0000 0000 | RP |
| <MTTCAN0_base> + $08C_H$ | | reserved (1) | | 0000 0000 | R |
| <MTTCAN0_base> + $090_H$ | MTTCAN0XIDAM | Extended ID AND Mask | 2949 | 1FFF FFFF | RP |
| <MTTCAN0_base> + $094_H$ | MTTCAN0HPMS | High Priority Message Status | 2950 | 0000 0000 | R |
| <MTTCAN0_base> + $098_H$ | MTTCAN0NDAT1 | New Data 1 | 2951 | 0000 0000 | RW |
| <MTTCAN0_base> + $09C_H$ | MTTCAN0NDAT2 | New Data 2 | 2952 | 0000 0000 | RW |
| <MTTCAN0_base> + $0A0_H$ | MTTCAN0RXF0C | Rx FIFO 0 Configuration | 2953 | 0000 0000 | RP |

Bosch CAN FD Specification V1.0

Table 37.68 M_TTCAN Register Map (2/2)

| ADDRESS | SYMBOL | NAME | PAGE | RESET | ACC |
|---|---|---|---|---|---|
| <MTTCAN0_base> + $0A4_H$ | MTTCAN0RXF0S | Rx FIFO 0 Status | 2954 | 0000 0000 | R |
| <MTTCAN0_base> + $0A8_H$ | MTTCAN0RXF0A | Rx FIFO 0 Acknowledge | 2955 | 0000 0000 | RW |
| <MTTCAN0_base> + $0AC_H$ | MTTCAN0RXBC | Rx Buffer Configuration | 2956 | 0000 0000 | RP |
| <MTTCAN0_base> + $0B0_H$ | MTTCAN0RXF1C | Rx FIFO 1 Configuration | 2957 | 0000 0000 | RP |
| <MTTCAN0_base> + $0B4_H$ | MTTCAN0RXF1S | Rx FIFO 1 Status | 2958 | 0000 0000 | R |
| <MTTCAN0_base> + $0B8_H$ | MTTCAN0RXF1A | Rx FIFO 1 Acknowledge | 2959 | 0000 0000 | RW |
| <MTTCAN0_base> + $0BC_H$ | MTTCAN0RXESC | Rx Buffer / FIFO Element Size Configuration | 2960 | 0000 0000 | RP |
| <MTTCAN0_base> + $0C0_H$ | MTTCAN0TXBC | Tx Buffer Configuration | 2961 | 0000 0000 | RP |
| <MTTCAN0_base> + $0C4_H$ | MTTCAN0TXFQS | Tx FIFO/Queue Status | 2962 | 0000 0000 | R |
| <MTTCAN0_base> + $0C8_H$ | MTTCAN0TXESC | Tx Buffer Element Size Configuration | 2963 | 0000 0000 | RP |
| <MTTCAN0_base> + $0CC_H$ | MTTCAN0TXBRP | Tx Buffer Request Pending | 2964 | 0000 0000 | R |
| <MTTCAN0_base> + $0D0_H$ | MTTCAN0TXBAR | Tx Buffer Add Request | 2965 | 0000 0000 | RW |
| <MTTCAN0_base> + $0D4_H$ | MTTCAN0TXBCR | Tx Buffer Cancellation Request | 2966 | 0000 0000 | RW |
| <MTTCAN0_base> + $0D8_H$ | MTTCAN0TXBTO | Tx Buffer Transmission Occurred | 2967 | 0000 0000 | R |
| <MTTCAN0_base> + $0DC_H$ | MTTCAN0TXBCF | Tx Buffer Cancellation Finished | 2968 | 0000 0000 | R |
| <MTTCAN0_base> + $0E0_H$ | MTTCAN0TXBTIE | Tx Buffer Transmission Interrupt Enable | 2968 | 0000 0000 | RW |
| <MTTCAN0_base> + $0E4_H$ | MTTCAN0TXBCIE | Tx Buffer Cancellation Finished Interrupt Enable | 2969 | 0000 0000 | RW |
| <MTTCAN0_base> + $0E8_H$ to $0EC_H$ | | reserved (2) | | 0000 0000 | R |
| <MTTCAN0_base> + $0F0_H$ | MTTCAN0TXEFC | Tx Event FIFO Configuration | 2970 | 0000 0000 | RP |
| <MTTCAN0_base> + $0F4_H$ | MTTCAN0TXEFS | Tx Event FIFO Status | 2971 | 0000 0000 | R |
| <MTTCAN0_base> + $0F8_H$ | MTTCAN0TXEFA | Tx Event FIFO Acknowledge | 2972 | 0000 0000 | RW |
| <MTTCAN0_base> + $0FC_H$ | | reserved (1) | | 0000 0000 | R |
| <MTTCAN0_base> + $100_H$ | MTTCAN0TTTMC | TT Trigger Memory Configuration | 2973 | 0000 0000 | RP |
| <MTTCAN0_base> + $104_H$ | MTTCAN0TTRMC | TT Reference Message Configuration | 2974 | 0000 0000 | RP |
| <MTTCAN0_base> + $108_H$ | MTTCAN0TTOCF | TT Operation Configuration | 2975 | 0001 0000 | RP |
| <MTTCAN0_base> + $10C_H$ | MTTCAN0TTMLM | TT Matrix Limits | 2977 | 0000 0000 | RP |
| <MTTCAN0_base> + $110_H$ | MTTCAN0TURCF | TUR Configuration | 2978 | 1000 0000 | RP |
| <MTTCAN0_base> + $114_H$ | MTTCAN0TTOCN | TT Operation Control | 2980 | 0000 0000 | RW |
| <MTTCAN0_base> + $118_H$ | MTTCAN0TTGTP | TT Global Time Preset | 2982 | 0000 0000 | RW |
| <MTTCAN0_base> + $11C_H$ | MTTCAN0TTTMK | TT Time Mark | 2983 | 0000 0000 | RW |
| <MTTCAN0_base> + $120_H$ | MTTCAN0TTIR | TT Interrupt Register | 2984 | 0000 0000 | RW |
| <MTTCAN0_base> + $124_H$ | MTTCAN0TTIE | TT Interrupt Enable | 2986 | 0000 0000 | RW |
| <MTTCAN0_base> + $128_H$ | MTTCAN0TTILS | TT Interrupt Line Select | 2987 | 0000 0000 | RW |
| <MTTCAN0_base> + $12C_H$ | MTTCAN0TTOST | TT Operation Status | 2988 | 0000 0080 | R |
| <MTTCAN0_base> + $130_H$ | MTTCAN0TURNA | TUR Numerator Actual | 2990 | 0001 0000 | R |
| <MTTCAN0_base> + $134_H$ | MTTCAN0TTLGT | TT Local & Global Time | 2991 | 0000 0000 | R |
| <MTTCAN0_base> + $138_H$ | MTTCAN0TTCTC | TT Cycle Time & Count | 2992 | 003F 0000 | R |
| <MTTCAN0_base> + $13C_H$ | MTTCAN0TTCPT | TT Capture Time | 2993 | 0000 0000 | R |
| <MTTCAN0_base> + $140_H$ | MTTCAN0TTCSM | TT Cycle Sync Mark | 2994 | 0000 0000 | R |
| <MTTCAN0_base> + $144_H$ to $1FC_H$ | | reserved (47) | | 0000 0000 | R |

**Note:** R = Read, S = Set on read, X = Reset on read, W = Write, P = Protected write, p = Protected set, C = Clear/preset on write

Bosch CAN FD Specification V1.0

### 37.6.2.3 Registers

#### (1) MTTCAN0CREL — Core Release Register

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $000_H$

**Value after reset:** $3013\ 0508_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | REL[3:0] | | | | STEP[3:0] | | | | SUBSTEP[3:0] | | | | YEAR[3:0] | | | |
| Value after reset | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MON[7:0] | | | | | | | | DAY[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.69 MTTCAN0CREL Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | REL[3:0] | Core Release<br>One digit, BCD-coded. |
| 27 to 24 | STEP[3:0] | Step of Core Release<br>One digit, BCD-coded. |
| 23 to 20 | SUBSTEP[3:0] | Sub-step of Core Release<br>One digit, BCD-coded. |
| 19 to 16 | YEAR[3:0] | Time Stamp Year<br>One digit, BCD-coded. This field is set by generic parameter on M_TTCAN synthesis. |
| 15 to 8 | MON[7:0] | Time Stamp Month<br>Two digits, BCD-coded. This field is set by generic parameter on M_TTCAN synthesis. |
| 7 to 0 | DAY[7:0] | Time Stamp Day<br>Two digits, BCD-coded. This field is set by generic parameter on M_TTCAN synthesis. |

**Table 37.70 Coding of Revisions**

| Release | Step | SubStep | Year | Month | Day | Name |
|---|---|---|---|---|---|---|
| 3 | 0 | 1 | 3 | 05 | 08 | Revision 3.0.1, Date 2013/05/08 |

### (2) MTTCAN0ENDN — Endian Register

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $004_H$

**Value after reset:** $8765\ 4321_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETV[31:16] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETV[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.71 MTTCAN0ENDN Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ETV[31:0] | Endianness Test Value<br>The endianness test value is $8765\ 4321_H$. |

Bosch CAN-FD Specification V1.0

### (3) MTTCAN0FBTP — Fast Bit Timing & Prescaler Register

This register is only writable if bits MTTCAN0CCCR.CCE and MTTCAN0CCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 m_ttcan_cclk (CLKP_H2) periods. tq = (FBRP + 1) m_ttcan_cclk (CLKP_H2) period.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. FTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [FTSEG1 + FTSEG2 + 3] tq
or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $00C_H$

**Value after reset:** $0000\ 0A33_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | TDCO[4:0] | | | | | TDC | — | — | FBRP[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | RP | RP | RP | RP | RP | RP | R | R | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | FTSEG1[3:0] | | | | — | FTSEG2[2:0] | | | — | — | FSJW[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | RP | RP | RP | RP | R | RP | RP | RP | R | R | RP | RP |

Bosch CAN-FD Specification V1.0

**Table 37.72 MTTCAN0FBTP Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 28 to 24 | TDCO[4:0] | Transmitter Delay Compensation Offset<br>$00_H$ to $1F_H$<br>Offset value defining the distance between the measured delay from m_ttcan_tx to m_ttcan_rx and the secondary sample point. Valid values are 0 to 31 m_ttcan_cclk (CLKP_H2) periods. |
| 23 | TDC | Transmitter Delay Compensation<br>0: Transmitter Delay Compensation disabled<br>1: Transmitter Delay Compensation enabled |
| 22, 21 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 20 to 16 | FBRP[4:0] | Fast Baud Rate Prescaler<br>$00_H$ to $1F_H$<br>The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 15 to 12 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 11 to 8 | FTSEG1[3:0] | Fast time segment before sample point<br>$1_H$ to $F_H$<br>Valid values are 1 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 7 | Reserved | These bits are always read as 0. When written, write the initial value. |

**Table 37.72 MTTCAN0FBTP Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 6 to 4 | FTSEG2[2:0] | Fast time segment after sample point<br>$0_H$ to $7_H$<br>Valid values are 0 to 7. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 3, 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1, 0 | FSJW[1:0] | Fast (Re) Synchronization Jump Width<br>$0_H$ to $3_H$<br>Valid values are 0 to 3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |

**NOTES**

1. With a CAN clock (m_ttcan_cclk: CLKP_H2) of 8 MHz, the reset value of 0000 $0A33_H$ configures the M_TTCAN for a fast bit rate of 500 kBit/s.
2. The bit rate configured for the CAN FD data phase via MTTCAN0FBTP must be higher or equal to the bit rate configured for the arbitration phase via MTTCAN0BTP.

Bosch CAN-FD Specification V1.0

### (4) MTTCAN0TEST — Test Register

Write access to the Test Register has to be enabled by setting bit MTTCAN0CCCR.TEST to '1'. All Test Register functions are set to their reset values when bit MTTCAN0CCCR.TEST is reset.

Loop Back Mode and software control of pin m_ttcan_tx are hardware test modes. Programming of TX [1] "00" may disturb the message transfer on the CAN bus.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $010_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TDCV[5:0] | | | | | | RX | TX[1:0] | | LBCK | CAT | CAM | TAT | TAM |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | RP | RP | RP | R | R | RP | RP |

Bosch E-AI-FD Spec ifica tion V1.0

**Table 37.73 MTTCAN0TEST Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 13 to 8 | TDCV[5:0] | Transmitter Delay Compensation Value<br>$00_H$ to $3F_H$<br>Position of the secondary sample point, defined by the sum of the measured delay from m_ttcan_tx to m_ttcan_rx and MTTCAN0FBTP.TDCO. Valid values are 0 to 63 m_ttcan_cclk (CLKP_H2) periods. |
| 7 | RX | Receive Pin<br>Monitors the actual value of pin m_ttcan_rx<br>0: The CAN bus is dominant (m_ttcan_rx = '0').<br>1: The CAN bus is recessive (m_ttcan_rx = '1'). |
| 6, 5 | TX[1:0] | Control of Transmit Pin<br>00: Reset value, m_ttcan_tx controlled by the CAN Core, updated at the end of the CAN bit time.<br>01: Sample Point can be monitored at pin m_ttcan_tx.<br>10: Dominant ('0') level at pin m_ttcan_tx<br>11: Recessive ('1') at pin m_ttcan_tx |
| 4 | LBCK | Loop Back Mode<br>0: Reset value, Loop Back Mode is disabled.<br>1: Loop Back Mode is enabled (see **(9) Test Modes**). |
| 3 | CAT | Check ASC Transmit Control<br>Monitors level at output pin m_ttcan_asct.<br>0: Output pin m_ttcan_asct = '0'<br>1: Output pin m_ttcan_asct = '1' |

Table 37.73 MTTCAN0TEST Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 2 | CAM | Check ASC Multiplexer Control<br>Monitors level at output pin m_ttcan_ascm.<br>0: Output pin m_ttcan_ascm = '0'<br>1: Output pin m_ttcan_ascm = '1' |
| 1 | TAT | Test ASC Transmit Control<br>Controls output pin m_ttcan_asct in test mode, ORed with the signal from the FSE<br>0: Level at pin m_ttcan_asct controlled by FSE<br>1: Level at pin m_ttcan_asct = '1' |
| 0 | TAM | Test ASC Multiplexer Control<br>Controls output pin m_ttcan_ascm in test mode, ORed with the signal from the FSE<br>0: Level at pin m_ttcan_ascm controlled by FSE<br>1: Level at pin m_ttcan_ascm = '1' |

### (5) MTTCAN0RWD — RAM Watchdog

The RAM Watchdog monitors the READY output of the Message RAM (m_ttcan_aeim_ready). A Message RAM access via the M_TTCAN's Generic Master Interface (m_ttcan_aeim_sel active) starts the Message RAM Watchdog Counter with the value configured by MTTCAN0RWD.WDC. The counter is reloaded with MTTCAN0RWD.WDC when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag MTTCAN0IR.WDI is set. The RAM Watchdog Counter is clocked by the Host clock (m_ttcan_hclk: CLK_HSB).

Bosch CAN-FD Specification V1.0

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $014_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WDV[7:0] | | | | | | | | WDC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP |

Table 37.74 MTTCAN0RWD Register Contents

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 8 | WDV[7:0] | Watchdog Value<br>Actual Message RAM Watchdog Counter Value. |
| 7 to 0 | WDC[7:0] | Watchdog Configuration<br>Start value of the Message RAM Watchdog Counter. With the reset value of "00" the counter is disabled. |

### (6) MTTCAN0CCCR — CC Control Register

For details about setting and resetting of single bits see **(1) Software Initialization**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $018_H$

**Value after reset:** $0000\ 0001_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TXP | FDBS | FDO | CMR[1:0] | | CME[1:0] | | TEST | DAR | MON | CSR | CSA | ASM | CCE | INIT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | RP | R | R | R/W | R/W | RP | RP | Rp | RP | Rp | R/W | R | Rp | RP | R/W |

**Table 37.75 MTTCAN0CCCR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 15 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 14 | TXP | Transmit Pause<br>If this bit is set, the M_TTCAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame (see **(1) Transmit Pause**).<br>0: Transmit pause disabled<br>1: Transmit pause enabled |
| 13 | FDBS | CAN FD Bit Rate Switching<br>0: This node transmits no frames with bit rate switching.<br>1: This node transmits all frames (excl. remote frames) with bit rate switching. |
| 12 | FDO | CAN FD Operation<br>0: This node transmits all frames in CAN format according to ISO11898-1.<br>1: This node transmits all frames (excl. remote frames) in CAN FD format. |
| 11, 10 | CMR[1:0] | CAN Mode Request<br>A change of the CAN operation mode is requested by writing to this bit field. After change to the requested operation mode the bit field is reset to “00” and the status flags FDBS and FDO are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CMR is retained until it is overwritten by the next mode change request. In case CME = “01”/”10”/”11” a change to CAN operation according to ISO 11898-1 is always possible. Default is CAN operation according to ISO11898-1.<br>00: Unchanged<br>01: Request CAN FD operation<br>10: Request CAN FD operation with bit rate switching<br>11: Request CAN operation according ISO11898-1 |
| 9, 8 | CME[1:0] | CAN Mode Enable<br>00: CAN operation according to ISO11898-1 enabled<br>01: CAN FD operation enabled<br>10: CAN FD operation with bit rate switching enabled<br>11: CAN FD operation with bit rate switching enabled<br>**NOTE**<br>When CME = “00”, received frames are strictly interpreted according to ISO11898-1, which leads to the transmission of an error frame when receiving a CAN FD frame. In case CME = “01”, transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CME = “10”/“11”, transmission and reception of long and fast CAN FD frames is enabled. |

Bosch MCAN Specification V1.0

**Table 37.75 MTTCAN0CCCR Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | TEST | Test Mode Enable<br>0: Normal operation, register MTTCAN0TEST holds reset values.<br>1: Test Mode, write access to register MTTCAN0TEST enabled. |
| 6 | DAR | Disable Automatic Retransmission<br>0: Automatic retransmission of messages not transmitted successfully enabled.<br>1: Automatic retransmission disabled |
| 5 | MON | Bus Monitoring Mode<br>Bit MON can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time.<br>0: Bus Monitoring Mode is disabled<br>1: Bus Monitoring Mode is enabled |
| 4 | CSR | Clock Stop Request<br>0: No clock stop is requested.<br>1: Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle. |
| 3 | CSA | Clock Stop Acknowledge<br>0: No clock stop acknowledged.<br>1: M_TTCAN may be set in power down by stopping m_ttcan_hclk (CLK_HSB) and m_ttcan_cclk (CLKP_H2). |
| 2 | ASM | Restricted Operation Mode<br>Bit ASM can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. For a description of the Restricted Operation Mode see **(5) Restricted Operation Mode**.<br>0: Normal CAN operation<br>1: Restricted Operation Mode active |
| 1 | CCE | Configuration Change Enable<br>0: The CPU has no write access to the protected configuration registers.<br>1: The CPU has write access to the protected configuration registers (while MTTCAN0CCCR.INIT = '1'). |
| 0 | INIT | Initialization<br>0: Normal Operation<br>1: Initialization is started. |

Bosch CAN FD Specification V1.0

**NOTE**

Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

### (7) MTTCAN0BTP — Bit Timing & Prescaler Register

This register is only writable if bits MTTCAN0CCCR.CCE and MTTCAN0CCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 81 time quanta. The CAN time quantum may be programmed in the range of 1 to 1024 m_ttcan_cclk (CLKP_H2) periods. tq = (BRP + 1) m_ttcan_cclk (CLKP_H2) period.

TSEG1 is the sum of Prop_Seg and Phase_Seg1. TSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [TSEG1 + TSEG2 + 3] tq
or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $01C_H$

**Value after reset:** 0000 $0A33_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | BRP[9:0] | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TSEG1[5:0] | | | | | | TSEG2[3:0] | | | | SJW[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| R/W | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

Bosch CAN-FD Specification V1.0

**Table 37.76 MTTCAN0BTP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 25 to 16 | BRP[9:0] | Baud Rate Prescaler<br>$000_H$ to $3FF_H$<br>The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 15, 14 | Reserved | These bits are always read as 0. |
| 13 to 8 | TSEG1[5:0] | Time segment before sample point<br>$01_H$ to $3F_H$<br>Valid values are 1 to 63. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 7 to 4 | TSEG2[3:0] | Time segment after sample point<br>$0_H$ to $F_H$<br>Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. |
| 3 to 0 | SJW[3:0] | (Re) Synchronization Jump Width<br>$0_H$ to $F_H$<br>Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |

**NOTE**

With a CAN clock (m_ttcan_cclk: CLKP_H2) of 8 MHz, the reset value of 0000 $0A33_H$ configures the M_TTCAN for a bit rate of 500 kBit/s.

### (8) MTTCAN0TSCC — Timestamp Counter Configuration

For a description of the Timestamp Counter see **Section 37.6.3.2, Timestamp Generation**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $020_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TCP[3:0] | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TSS[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP |

**Table 37.77 MTTCAN0TSCC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 20 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 19 to 16 | TCP[3:0] | Timestamp Counter Prescaler<br>$0_H$ to $F_H$<br>Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 to 16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.<br>**NOTE**<br>With CAN FD an external counter is required for timestamp generation (TSS = “10”) |
| 15 to 2 | Reserved | These bits are always read as 0. |
| 1, 0 | TSS[1:0] | Timestamp Select<br>00: Timestamp counter value always $0000_H$<br>01: Timestamp counter value incremented according to TCP<br>10: External timestamp counter value used<br>11: Same as “00” |

Bosch CAN FD Specification V1.0

### (9) MTTCAN0TSCV — Timestamp Counter Value

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $024_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSC[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

**Table 37.78 MTTCAN0TSCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 0 | TSC[15:0] | Timestamp Counter<br>The internal/external Timestamp Counter value is captured on start of frame (both Rx and Tx). When MTTCAN0TSCC.TSS = “01”, the Timestamp Counter is incremented in multiples of CAN bit times [1 to 16] depending on the configuration of MTTCAN0TSCC.TCP. A wrap around sets interrupt flag MTTCAN0IR.TSW. Write access resets the counter to zero. When MTTCAN0TSCC.TSS = “10”, TSC reflects the external Timestamp Counter value. A write access has no impact. |

**NOTE**

A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by write access to MTTCAN0TSCV.

Closed CANFD Specification Ver. 0

### (10) MTTCAN0TOCC — Timeout Counter Configuration

For a description of the Timeout Counter see **Section 37.6.3.3, Timeout Counter**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $028_H$

**Value after reset:** FFFF $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TOP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TOS[1:0] | | ETOC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP |

**Table 37.79 MTTCAN0TOCC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | TOP[15:0] | Timeout Period<br>Start value of the Timeout Counter (down-counter). Configures the Timeout Period. |
| 15 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2, 1 | TOS[1:0] | Timeout Select<br>When operating in Continuous mode, a write to MTTCAN0TOCV presets the counter to the value configured by MTTCAN0TOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by MTTCAN0TOCC.TOP. Down-counting is started when the first FIFO element is stored.<br>00: Continuous operation<br>01: Timeout controlled by Tx Event FIFO<br>10: Timeout controlled by Rx FIFO 0<br>11: Timeout controlled by Rx FIFO 1 |
| 0 | ETOC | Enable Timeout Counter<br>0: Timeout Counter disabled<br>1: Timeout Counter enabled |

**NOTE**

For use of timeout function with CAN FD see **Section 37.5.3.3, Timeout Counter**.

Bosch CAN-FD Specification V1.0

### (11) MTTCAN0TOCV — Timeout Counter Value

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $02C_H$

**Value after reset:** $0000\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TOC[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

**Table 37.80 MTTCAN0TOCV Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 0 | TOC[15:0] | Timeout Counter<br>The Timeout Counter is decremented in multiples of CAN bit times [1 to 16] depending on the configuration of MTTCAN0TSCC.TCP. When decremented to zero, interrupt flag MTTCAN0IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via MTTCAN0TOCC.TOS. |

Bosch CAN FD Specification V1.0

### (12) MTTCAN0ECR — Error Counter Register

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $040_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CEL[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | X | X | X | X | X | X | X | X |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RP | REC[6:0] | | | | | | | TEC[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.81 MTTCAN0ECR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. |
| 23 to 16 | CEL[7:0] | CAN Error Logging<br>The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at $FF_H$; the next increment of TEC or REC sets interrupt flag MTTCAN0IR.ELO. |
| 15 | RP | Receive Error Passive<br>0: The Receive Error Counter is below the error passive level of 128<br>1: The Receive Error Counter has reached the error passive level of 128 |
| 14 to 8 | REC[6:0] | Receive Error Counter<br>Actual state of the Receive Error Counter, values between 0 and 127 |
| 7 to 0 | TEC[7:0] | Transmit Error Counter<br>Actual state of the Transmit Error Counter, values between 0 and 255 |

Boot CANFD Specification V1.0

**NOTE**

When MTTCAN0CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented. This enables monitoring of collisions between CAN frames and ASC frames.

### (13) MTTCAN0PSR — Protocol Status Register

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $044_H$

**Value after reset:** 0000 $0707_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | REDL | RBRS | RESI | FLEC[2:0] | | | BO | EW | EP | ACT[1:0] | | LEC[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| R/W | R | R | X | X | X | S | S | S | R | R | R | R | R | S | S | S |

**Table 37.82 MTTCAN0PSR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 14 | Reserved | These bits are always read as 0. |
| 13 | REDL | Received a CAN FD Message<br>This bit is set independent of acceptance filtering.<br>0: Since this bit was reset by the CPU, no CAN FD message has been received<br>1: Message in CAN FD format with EDL flag set has been received |
| 12 | RBRS | BRS flag of last received CAN FD Message<br>This bit is set together with REDL, independent of acceptance filtering.<br>0: Last received CAN FD message did not have its BRS flag set<br>1: Last received CAN FD message had its BRS flag set |
| 11 | RESI | ESI flag of last received CAN FD Message<br>This bit is set together with REDL, independent of acceptance filtering.<br>0: Last received CAN FD message did not have its ESI flag set<br>1: Last received CAN FD message had its ESI flag set |
| 10 to 8 | FLEC[2:0] | Fast Last Error Code<br>Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. |
| 7 | BO | Bus_Off Status<br>0: The M_TTCAN is not Bus_Off<br>1: The M_TTCAN is in Bus_Off state |
| 6 | EW | Warning Status<br>0: Both error counters are below the Error_Warning limit of 96<br>1: At least one of error counter has reached the Error_Warning limit of 96 |

Bosch CAN FD Specification V1.0

Table 37.82 MTTCAN0PSR Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 5 | EP | Error Passive<br>0: The M_TTCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected<br>1: The M_TTCAN is in the Error_Passive state |
| 4, 3 | ACT[1:0] | Activity<br>Monitors the module's CAN communication state.<br>00: Synchronizing - node is synchronizing on CAN communication<br>01: Idle - node is neither receiver nor transmitter<br>10: Receiver - node is operating as receiver<br>11: Transmitter - node is operating as transmitter |
| 2 to 0 | LEC[2:0] | Last Error Code<br>The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error.<br>0: No Error: No error occurred since LEC has been reset by successful reception or transmission.<br>1: Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.<br>2: Form Error: A fixed format part of a received frame has the wrong format.<br>3: AckError: The message transmitted by the M_TTCAN was not acknowledged by another node.<br>4: Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.<br>5: Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).<br>6: CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.<br>7: NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register. |

Bosch CAN IP
Specification V1.0

NOTES

1. When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.
2. The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting MTTCAN0CCCR.INIT. If the device goes Bus_Off, it will set MTTCAN0CCCR.INIT of its own accord, stopping all bus activities. Once MTTCAN0CCCR.INIT has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of MTTCAN0CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to MTTCAN0PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. MTTCAN0ECR.REC is used to count these sequences.

### (14) MTTCAN0IR — Interrupt Register

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. The configuration of MTTCAN0IE controls whether an interrupt is generated. The configuration of MTTCAN0ILS controls on which interrupt line an interrupt is signalled.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $050_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STE | FOE | ACKE | BE | CRCE | WDI | BO | EW | EP | ELO | BEU | BEC | DRX | TOO | MRAF | TSW |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFL | TEFF | TEFW | TEFN | TFE | TCF | TC | HPM | RF1L | RF1F | RF1W | RF1N | RF0L | RF0F | RF0W | RF0N |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bosch CAN-FD Specification V1.0

Table 37.83 MTTCAN0IR Register Contents (1/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | STE | Stuff Error<br>0: No Stuff Error detected<br>1: More than 5 equal bits in a sequence occurred |
| 30 | FOE | Format Error<br>0: No Format Error detected<br>1: A fixed format part of a received frame has the wrong format |
| 29 | ACKE | Acknowledge Error<br>0: No Acknowledge Error detected<br>1: A transmitted message was not acknowledged by another node |
| 28 | BE | Bit Error<br>0: No Bit Error detected<br>1: Device wanted to send a rec / dom level, but monitored bus level was dom / rec |
| 27 | CRCE | CRC Error<br>0: No CRC Error detected<br>1: Received CRC did not match the calculated CRC |
| 26 | WDI | Watchdog Interrupt<br>0: No Message RAM Watchdog event occurred<br>1: Message RAM Watchdog event due to missing READY |
| 25 | BO | Bus_Off Status<br>0: Bus_Off status unchanged<br>1: Bus_Off status changed |
| 24 | EW | Warning Status<br>0: Error_Warning status unchanged<br>1: Error_Warning status changed |
| 23 | EP | Error Passive<br>0: Error_Passive status unchanged<br>1: Error_Passive status changed |
| 22 | ELO | Error Logging Overflow<br>0: CAN Error Logging Counter did not overflow<br>1: Overflow of CAN Error Logging Counter occurred |

Table 37.83 MTTCAN0IR Register Contents (2/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 21 | BEU | Bit Error Uncorrected<br>Message RAM bit error detected, uncorrected. Controlled by input signal m_ttcan_aeim_berr[1] generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets MTTCAN0CCCR.INIT to '1'. This is done to avoid transmission of corrupted data.<br>0: No bit error detected when reading from Message RAM<br>1: Bit error detected, uncorrected (e.g. parity logic) |
| 20 | BEC | Bit Error Corrected<br>Message RAM bit error detected and corrected. Controlled by input signal m_ttcan_aeim_berr[0] generated by an optional external parity / ECC logic attached to the Message RAM.<br>0: No bit error detected when reading from Message RAM<br>1: Bit error detected and corrected (e.g. ECC) |
| 19 | DRX | Message stored to Dedicated Rx Buffer<br>The flag is set whenever a received message has been stored into a dedicated Rx Buffer.<br>0: No Rx Buffer updated<br>1: At least one received message stored into an Rx Buffer |
| 18 | TOO | Timeout Occurred<br>0: No timeout<br>1: Timeout reached |
| 17 | MRAF | Message RAM Access Failure<br>The flag is set, when the Rx Handler<br>• has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message.<br>• was not able to write a message to the Message RAM. In this case message storage is aborted.<br>In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location.<br>The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the M_TTCAN is switched into Restricted Operation Mode (see **(5) Restricted Operation Mode**). To leave Restricted Operation Mode, the Host CPU has to reset MTTCAN0CCCR.ASM.<br>0: No Message RAM access failure occurred<br>1: Message RAM access failure occurred |
| 16 | TSW | Timestamp Wraparound<br>0: No timestamp counter wrap-around<br>1: Timestamp counter wrapped around |
| 15 | TEFL | Tx Event FIFO Element Lost<br>0: No Tx Event FIFO element lost<br>1: Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero |
| 14 | TEFF | Tx Event FIFO Full<br>0: Tx Event FIFO not full<br>1: Tx Event FIFO full |
| 13 | TEFW | Tx Event FIFO Watermark Reached<br>0: Tx Event FIFO fill level below watermark<br>1: Tx Event FIFO fill level reached watermark |
| 12 | TEFN | Tx Event FIFO New Entry<br>0: Tx Event FIFO unchanged<br>1: Tx Handler wrote Tx Event FIFO element |
| 11 | TFE | Tx FIFO Empty<br>0: Tx FIFO non-empty<br>1: Tx FIFO empty |

Bosch CANFD Specification V1.0

Table 37.83 MTTCAN0IR Register Contents (3/3)

| Bit Position | Bit Name | Function |
|---|---|---|
| 10 | TCF | Transmission Cancellation Finished<br>0: No transmission cancellation finished<br>1: Transmission cancellation finished |
| 9 | TC | Transmission Completed<br>0: No transmission completed<br>1: Transmission completed |
| 8 | HPM | High Priority Message<br>0: No high priority message received<br>1: High priority message received |
| 7 | RF1L | Rx FIFO 1 Message Lost<br>0: No Rx FIFO 1 message lost<br>1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero |
| 6 | RF1F | Rx FIFO 1 Full<br>0: Rx FIFO 1 not full<br>1: Rx FIFO 1 full |
| 5 | RF1W | Rx FIFO 1 Watermark Reached<br>0: Rx FIFO 1 fill level below watermark<br>1: Rx FIFO 1 fill level reached watermark |
| 4 | RF1N | Rx FIFO 1 New Message<br>0: No new message written to Rx FIFO 1<br>1: New message written to Rx FIFO 1 |
| 3 | RF0L | Rx FIFO 0 Message Lost<br>0: No Rx FIFO 0 message lost<br>1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero |
| 2 | RF0F | Rx FIFO 0 Full<br>0: Rx FIFO 0 not full<br>1: Rx FIFO 0 full |
| 1 | RF0W | Rx FIFO 0 Watermark Reached<br>0: Rx FIFO 0 fill level below watermark<br>1: Rx FIFO 0 fill level reached watermark |
| 0 | RF0N | Rx FIFO 0 New Message<br>0: No new message written to Rx FIFO 0<br>1: New message written to Rx FIFO 0 |

Bosch CAN FD Specification V1.0

### (15) MTTCAN0IE — Interrupt Enable

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signalled on an interrupt line.

0: Interrupt disabled

1: Interrupt enabled

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $054_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STEE | FOEE | ACKEE | BEE | CRCEE | WDIE | BOE | EWE | EPE | ELOE | BEUE | BECE | DRXE | TOOE | MRAFE | TSWE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFLE | TEFFE | TEFWE | TEFNE | TFEE | TCFE | TCE | HPME | RF1LE | RF1FE | RF1WE | RF1NE | RF0LE | RF0FE | RF0WE | RF0NE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bosch CAN-FD Specification V1.0

**Table 37.84 MTTCAN0IE Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | STEE | Stuff Error Interrupt Enable |
| 30 | FOEE | Format Error Interrupt Enable |
| 29 | ACKEE | Acknowledge Error Interrupt Enable |
| 28 | BEE | Bit Error Interrupt Enable |
| 27 | CRCEE | CRC Error Interrupt Enable |
| 26 | WDIE | Watchdog Interrupt Enable |
| 25 | BOE | Bus_Off Status Interrupt Enable |
| 24 | EWE | Warning Status Interrupt Enable |
| 23 | EPE | Error Passive Interrupt Enable |
| 22 | ELOE | Error Logging Overflow Interrupt Enable |
| 21 | BEUE | Bit Error Uncorrected Interrupt Enable |
| 20 | BECE | Bit Error Corrected Interrupt Enable |
| 19 | DRXE | Message stored to Dedicated Rx Buffer Interrupt Enable |
| 18 | TOOE | Timeout Occurred Interrupt Enable |
| 17 | MRAFE | Message RAM Access Failure Interrupt Enable |
| 16 | TSWE | Timestamp Wraparound Interrupt Enable |
| 15 | TEFLE | Tx Event FIFO Event Lost Interrupt Enable |
| 14 | TEFFE | Tx Event FIFO Full Interrupt Enable |
| 13 | TEFWE | Tx Event FIFO Watermark Reached Interrupt Enable |
| 12 | TEFNE | Tx Event FIFO New Entry Interrupt Enable |
| 11 | TFEE | Tx FIFO Empty Interrupt Enable |
| 10 | TCFE | Transmission Cancellation Finished Interrupt Enable |
| 9 | TCE | Transmission Completed Interrupt Enable |
| 8 | HPME | High Priority Message Interrupt Enable |

Table 37.84 MTTCAN0IE Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 7 | RF1LE | Rx FIFO 1 Message Lost Interrupt Enable |
| 6 | RF1FE | Rx FIFO 1 Full Interrupt Enable |
| 5 | RF1WE | Rx FIFO 1 Watermark Reached Interrupt Enable |
| 4 | RF1NE | Rx FIFO 1 New Message Interrupt Enable |
| 3 | RF0LE | Rx FIFO 0 Message Lost Interrupt Enable |
| 2 | RF0FE | Rx FIFO 0 Full Interrupt Enable |
| 1 | RF0WE | Rx FIFO 0 Watermark Reached Interrupt Enable |
| 0 | RF0NE | Rx FIFO 0 New Message Interrupt Enable |

Bosch CAN-FD Specification V1.0

### (16) MTTCAN0ILS — Interrupt Line Select

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via MTTCAN0ILE.EINT0 and MTTCAN0ILE.EINT1.

0: Interrupt assigned to interrupt line m_ttcan_int0 (INTMTTCANI0)

1: Interrupt assigned to interrupt line m_ttcan_int1 (INTMTTCANI1)

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $058_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STEL | FOEL | ACKEL | BEL | CRCEL | WDIL | BOL | EWL | EPL | ELOL | BEUL | BECL | DRXL | TOOL | MRAFL | TSWL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TEFLL | TEFFL | TEFWL | TEFNL | TFEL | TCFL | TCL | HPML | RF1LL | RF1FL | RF1WL | RF1NL | RF0LL | RF0FL | RF0WL | RF0NL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bosch CAN-FD Specification V1.0

**Table 37.85 MTTCAN0ILS Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | STEL | Stuff Error Interrupt Line |
| 30 | FOEL | Format Error Interrupt Line |
| 29 | ACKEL | Acknowledge Error Interrupt Line |
| 28 | BEL | Bit Error Interrupt Line |
| 27 | CRCEL | CRC Error Interrupt Line |
| 26 | WDIL | Watchdog Interrupt Line |
| 25 | BOL | Bus_Off Status Interrupt Line |
| 24 | EWL | Warning Status Interrupt Line |
| 23 | EPL | Error Passive Interrupt Line |
| 22 | ELOL | Error Logging Overflow Interrupt Line |
| 21 | BEUL | Bit Error Uncorrected Interrupt Line |
| 20 | BECL | Bit Error Corrected Interrupt Line |
| 19 | DRXL | Message stored to Dedicated Rx Buffer Interrupt Line |
| 18 | TOOL | Timeout Occurred Interrupt Line |
| 17 | MRAFL | Message RAM Access Failure Interrupt Line |
| 16 | TSWL | Timestamp Wraparound Interrupt Line |
| 15 | TEFLL | Tx Event FIFO Event Lost Interrupt Line |
| 14 | TEFFL | Tx Event FIFO Full Interrupt Line |
| 13 | TEFWL | Tx Event FIFO Watermark Reached Interrupt Line |
| 12 | TEFNL | Tx Event FIFO New Entry Interrupt Line |
| 11 | TFEL | Tx FIFO Empty Interrupt Line |
| 10 | TCFL | Transmission Cancellation Finished Interrupt Line |
| 9 | TCL | Transmission Completed Interrupt Line |

Table 37.85 MTTCAN0ILS Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 8 | HPML | High Priority Message Interrupt Line |
| 7 | RF1LL | Rx FIFO 1 Message Lost Interrupt Line |
| 6 | RF1FL | Rx FIFO 1 Full Interrupt Line |
| 5 | RF1WL | Rx FIFO 1 Watermark Reached Interrupt Line |
| 4 | RF1NL | Rx FIFO 1 New Message Interrupt Line |
| 3 | RF0LL | Rx FIFO 0 Message Lost Interrupt Line |
| 2 | RF0FL | Rx FIFO 0 Full Interrupt Line |
| 1 | RF0WL | Rx FIFO 0 Watermark Reached Interrupt Line |
| 0 | RF0NL | Rx FIFO 0 New Message Interrupt Line |

Bosch CAN-FD Specification V1.0

### (17) MTTCAN0ILE — Interrupt Line Enable

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $05C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EINT1 | EINT0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

**Table 37.86 MTTCAN0ILE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 2 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 1 | EINT1 | Enable Interrupt Line 1<br>0: Interrupt line m_ttcan_int1 (INTMTTCANI1) disabled<br>1: Interrupt line m_ttcan_int1 (INTMTTCANI1) enabled |
| 0 | EINT0 | Enable Interrupt Line 0<br>0: Interrupt line m_ttcan_int0 (INTMTTCANI0) disabled<br>1: Interrupt line m_ttcan_int0 (INTMTTCANI0) enabled |

Bosch CAN FD Specification V1.0

### (18) MTTCAN0GFC — Global Filter Configuration

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in **Figure 37.19, Standard Message ID Filter Path** and **Figure 37.20, Extended Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $080_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ANFS[1:0] | | ANFE[1:0] | | RRFS | RRFE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP |

**Table 37.87 MTTCAN0GFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5, 4 | ANFS[1:0] | Accept Non-matching Frames Standard<br>Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.<br>00: Accept in Rx FIFO 0<br>01: Accept in Rx FIFO 1<br>10: Reject<br>11: Reject |
| 3, 2 | ANFE[1:0] | Accept Non-matching Frames Extended<br>Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.<br>00: Accept in Rx FIFO 0<br>01: Accept in Rx FIFO 1<br>10: Reject<br>11: Reject |
| 1 | RRFS | Reject Remote Frames Standard<br>0: Filter remote frames with 11-bit standard IDs<br>1: Reject all remote frames with 11-bit standard IDs |
| 0 | RRFE | Reject Remote Frames Extended<br>0: Filter remote frames with 29-bit extended IDs<br>1: Reject all remote frames with 29-bit extended IDs |

Bosch CAN FD Spec Rev 1.0

### (19) MTTCAN0SIDFC — Standard ID Filter Configuration

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in **Figure 37.19, Standard Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $084_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | LSS[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLSSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.88 MTTCAN0SIDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 24 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 23 to 16 | LSS[7:0] | List Size Standard<br>0: No standard Message ID filter<br>1 to 128: Number of standard Message ID filter elements<br>>128: Values greater than 128 are interpreted as 128 |
| 15 to 2 | FLSSA[15:2] | Filter List Standard Start Address<br>Start address of standard Message ID filter list (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

Boot CAN-TD Specification Ver.0

### (20) MTTCAN0XIDFC — Extended ID Filter Configuration

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in **Figure 37.20, Extended Message ID Filter Path**.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $088_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | LSE[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLESA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.89 MTTCAN0XIDFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 23 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 22 to 16 | LSE[6:0] | List Size Extended<br>0: No extended Message ID filter<br>1 to 64: Number of extended Message ID filter elements<br>>64: Values greater than 64 are interpreted as 64 |
| 15 to 2 | FLESA[15:2] | Filter List Extended Start Address<br>Start address of extended Message ID filter list (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

Bosch CAN FD Specification V1.0

### (21) MTTCAN0XIDAM — Extended ID AND Mask

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $090_H$

**Value after reset:** $1FFF\ FFFF_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | EIDM[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EIDM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

**Table 37.90 MTTCAN0XIDAM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 29 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 28 to 0 | EIDM[28:0] | Extended ID Mask<br>For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active. |

Bosch CAN FD Specification V1.0

### (22) MTTCAN0HPMS — High Priority Message Status

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $094_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLST | FIDX[6:0] | | | | | | | MSI[1:0] | | BIDX[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.91 MTTCAN0HPMS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. |
| 15 | FLST | Filter List<br>Indicates the filter list of the matching filter element.<br>0: Standard Filter List<br>1: Extended Filter List |
| 14 to 8 | FIDX[6:0] | Filter Index<br>Index of matching filter element. Range is 0 to MTTCAN0SIDFC.LSS – 1 resp. MTTCAN0XIDFC.LSE – 1. |
| 7, 6 | MSI[1:0] | Message Storage Indicator<br>00: No FIFO selected<br>01: FIFO message lost<br>10: Message stored in FIFO 0<br>11: Message stored in FIFO 1 |
| 5 to 0 | BIDX[5:0] | Buffer Index<br>Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = ‘1’. |

Bosch CAN FD Specification V1.0

### (23) MTTCAN0NDAT1 — New Data 1

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $098_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND31 | ND30 | ND29 | ND28 | ND27 | ND26 | ND25 | ND24 | ND23 | ND22 | ND21 | ND20 | ND19 | ND18 | ND17 | ND16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND15 | ND14 | ND13 | ND12 | ND11 | ND10 | ND9 | ND8 | ND7 | ND6 | ND5 | ND4 | ND3 | ND2 | ND1 | ND0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.92 MTTCAN0NDAT1 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ND[31:0] | New Data<br>The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.<br>0: Rx Buffer not updated<br>1: Rx Buffer updated from new message |

Bosch CAN FD Specification (ISO)

### (24) MTTCAN0NDAT2 — New Data 2

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $09C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND63 | ND62 | ND61 | ND60 | ND59 | ND58 | ND57 | ND56 | ND55 | ND54 | ND53 | ND52 | ND51 | ND50 | ND49 | ND48 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ND47 | ND46 | ND45 | ND44 | ND43 | ND42 | ND41 | ND40 | ND39 | ND38 | ND37 | ND36 | ND35 | ND34 | ND33 | ND32 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.93 MTTCAN0NDAT2 Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | ND[63:32] | New Data<br>The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.<br>0: Rx Buffer not updated<br>1: Rx Buffer updated from new message |

Bosch CAN FD Specification (ISO)

### (25) MTTCAN0RXF0C — Rx FIFO 0 Configuration

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0A0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F0OM | F0WM[6:0] | | | | | | | — | F0S[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F0SA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.94 MTTCAN0RXF0C Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | F0OM | FIFO 0 Operation Mode<br>FIFO 0 can be operated in blocking or in overwrite mode (see **(2) Rx FIFOs**).<br>0: FIFO 0 blocking mode<br>1: FIFO 0 overwrite mode |
| 30 to 24 | F0WM[6:0] | Rx FIFO 0 Watermark<br>0: Watermark interrupt disabled<br>1 to 64: Level for Rx FIFO 0 watermark interrupt (MTTCAN0IR.RF0W)<br>>64: Watermark interrupt disabled |
| 23 | Reserved | This bit is always read as 0. |
| 22 to 16 | F0S[6:0] | Rx FIFO 0 Size<br>0: No Rx FIFO 0<br>1 to 64: Number of Rx FIFO 0 elements<br>>64: Values greater than 64 are interpreted as 64<br>The Rx FIFO 0 elements are indexed from 0 to F0S-1 |
| 15 to 2 | F0SA[15:2] | Rx FIFO 0 Start Address<br>Start address of Rx FIFO 0 in Message RAM (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. |

Bosch CAN FD Specification V1.0

### (26) MTTCAN0RXF0S — Rx FIFO 0 Status

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0A4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RF0L | F0F | — | — | F0PI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | F0GI[5:0] | | | | | | — | F0FL[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.95 MTTCAN0RXF0S Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. |
| 25 | RF0L | Rx FIFO 0 Message Lost<br>This bit is a copy of interrupt flag MTTCAN0IR.RF0L. When MTTCAN0IR.RF0L is reset, this bit is also reset.<br>0: No Rx FIFO 0 message lost<br>1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero<br>**NOTE**<br>Overwriting the oldest message when MTTCAN0RXF0C.F0OM = '1' will not set this flag. |
| 24 | F0F | Rx FIFO 0 Full<br>0: Rx FIFO 0 not full<br>1: Rx FIFO 0 full |
| 23, 22 | Reserved | These bits are always read as 0. |
| 21 to 16 | F0PI[5:0] | Rx FIFO 0 Put Index<br>Rx FIFO 0 write index pointer, range 0 to 63. |
| 15, 14 | Reserved | These bits are always read as 0. |
| 13 to 8 | F0GI[5:0] | Rx FIFO 0 Get Index<br>Rx FIFO 0 read index pointer, range 0 to 63. |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 0 | F0FL[6:0] | Rx FIFO 0 Fill Level<br>Number of elements stored in Rx FIFO 0, range 0 to 64. |

Bosch CAN FD Specification V1.0

### (27) MTTCAN0RXF0A — Rx FIFO 0 Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0A8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | F0AI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.96 MTTCAN0RXF0A Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 to 0 | F0AI[5:0] | Rx FIFO 0 Acknowledge Index<br>After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index MTTCAN0RXF0S.F0GI to F0AI + 1 and update the FIFO 0 Fill Level MTTCAN0RXF0S.F0FL. |

Bosch CAN FD Specification V1.0

### (28) MTTCAN0RXBC — Rx Buffer Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0AC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RBSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.97 MTTCAN0RXBC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 to 2 | RBSA[15:2] | Rx Buffer Start Address<br>Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C. |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

Bosch CAN FD Specification V1.0

### (29) MTTCAN0RXF1C — Rx FIFO 1 Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0B0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1OM | F1WM[6:0] | | | | | | | — | F1S[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1SA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.98 MTTCAN0RXF1C Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | F1OM | FIFO 1 Operation Mode<br>FIFO 1 can be operated in blocking or in overwrite mode (see **(2) Rx FIFOs**).<br>0: FIFO 1 blocking mode<br>1: FIFO 1 overwrite mode |
| 30 to 24 | F1WM[6:0] | Rx FIFO 1 Watermark<br>0: Watermark interrupt disabled<br>1 to 64: Level for Rx FIFO 1 watermark interrupt (MTTCAN0IR.RF1W)<br>>64: Watermark interrupt disabled |
| 23 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 22 to 16 | F1S[6:0] | Rx FIFO 1 Size<br>0: No Rx FIFO 1<br>1 to 64: Number of Rx FIFO 1 elements<br>>64: Values greater than 64 are interpreted as 64<br>The Rx FIFO 1 elements are indexed from 0 to F1S - 1 |
| 15 to 2 | F1SA[15:2] | Rx FIFO 1 Start Address<br>Start address of Rx FIFO 1 in Message RAM (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

Bosch CAN FD Specification V1.0

### (30) MTTCAN0RXF1S — Rx FIFO 1 Status

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0B4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMS[1:0] | | — | — | — | — | RF1L | F1F | — | — | F1PI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | F1GI[5:0] | | | | | | — | F1FL[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.99 MTTCAN0RXF1S Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | DMS[1:0] | Debug Message Status<br>00: Idle state, wait for reception of debug messages, DMA request is cleared<br>01: Debug message A received<br>10: Debug messages A, B received<br>11: Debug messages A, B, C received, DMA request is set |
| 29 to 26 | Reserved | These bits are always read as 0. |
| 25 | RF1L | Rx FIFO 1 Message Lost<br>This bit is a copy of interrupt flag MTTCAN0IR.RF1L. When MTTCAN0IR.RF1L is reset, this bit is also reset.<br>0: No Rx FIFO 1 message lost<br>1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero<br>**NOTE**<br>Overwriting the oldest message when MTTCAN0RXF0C.F0OM = '1' will not set this flag. |
| 24 | F1F | Rx FIFO 1 Full<br>0: Rx FIFO 1 not full<br>1: Rx FIFO 1 full |
| 23, 22 | Reserved | These bits are always read as 0. |
| 21 to 16 | F1PI[5:0] | Rx FIFO 1 Put Index<br>Rx FIFO 1 write index pointer, range 0 to 63. |
| 15, 14 | Reserved | These bits are always read as 0. |
| 13 to 8 | F1GI[5:0] | Rx FIFO 1 Get Index<br>Rx FIFO 1 read index pointer, range 0 to 63. |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 0 | F1FL[6:0] | Rx FIFO 1 Fill Level<br>Number of elements stored in Rx FIFO 1, range 0 to 64. |

Bosch CAN FD Specification 1.0

### (31) MTTCAN0RXF1A — Rx FIFO 1 Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0B8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | F1AI[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.100 MTTCAN0RXF1A Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 6 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 5 to 0 | F1AI[5:0] | Rx FIFO 1 Acknowledge Index<br>After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index MTTCAN0RXF1S.F1GI to F1AI + 1 and update the FIFO 1 Fill Level MTTCAN0RXF1S.F1FL. |

Bosch CAN FD Specification V1.0

### (32) MTTCAN0RXESC — Rx Buffer / FIFO Element Size Configuration

Configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0BC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RBDS[2:0] | | | — | F1DS[2:0] | | | — | F0DS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | RP | RP | RP | R | RP | RP | RP | R | RP | RP | RP |

**Table 37.101 MTTCAN0RXESC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 11 | Reserved | These bits are always read as 0. |
| 10 to 8 | RBDS[2:0] | Rx Buffer Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |
| 7 | Reserved | This bit is always read as 0. |
| 6 to 4 | F1DS[2:0] | Rx FIFO 1 Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |
| 3 | Reserved | This bit is always read as 0. |
| 2 to 0 | F0DS[2:0] | Rx FIFO 0 Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |

**NOTE**

In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by MTTCAN0RXESC are stored to the Rx Buffer resp. Rx FIFO element. The rest of the frame's data field is ignored.

Bosch M_CAN IP Specification V3.2

### (33) MTTCAN0TXBC — Tx Buffer Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0C0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TFQM | TFQS[5:0] | | | | | | — | — | NDTB[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | RP | RP | RP | RP | RP | RP | RP | R | R | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.102 MTTCAN0TXBC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 30 | TFQM | Tx FIFO/Queue Mode<br>0: Tx FIFO operation<br>1: Tx Queue operation |
| 29 to 24 | TFQS[5:0] | Transmit FIFO/Queue Size<br>0: No Tx FIFO/Queue<br>1 to 32: Number of Tx Buffers used for Tx FIFO/Queue<br>>32: Values greater than 32 are interpreted as 32 |
| 23, 22 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 21 to 16 | NDTB[5:0] | Number of Dedicated Transmit Buffers<br>0: No Dedicated Tx Buffers<br>1 to 32: Number of Dedicated Tx Buffers<br>>32: Values greater than 32 are interpreted as 32 |
| 15 to 2 | TBSA[15:2] | Tx Buffers Start Address<br>Start address of Tx Buffers section in Message RAM (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

**NOTE**

Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

Bosch CAN FD Specification V1.0

### (34) MTTCAN0TXFQS — Tx FIFO/Queue Status

The Tx FIFO/Queue status is related to the pending Tx requests listed in register MTTCAN0TXBRP. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (MTTCAN0TXBRP not yet updated).

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0C4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | TFQF | TFQPI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | TFGI[4:0] | | | | | — | — | TFFL[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.103 MTTCAN0TXFQS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | These bits are always read as 0. |
| 21 | TFQF | Tx FIFO/Queue Full<br>0: Tx FIFO/Queue not full<br>1: Tx FIFO/Queue full |
| 20 to 16 | TFQPI[4:0] | Tx FIFO/Queue Put Index<br>Tx FIFO/Queue write index pointer, range 0 to 31. |
| 15 to 13 | Reserved | These bits are always read as 0. |
| 12 to 8 | TFGI[4:0] | Tx FIFO Get Index<br>Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (MTTCAN0TXBC.TFQM = '1'). |
| 7, 6 | Reserved | These bits are always read as 0. |
| 5 to 0 | TFFL[5:0] | Tx FIFO Free Level<br>Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (MTTCAN0TXBC.TFQM = '1') |

**NOTE**

In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers.
Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

Bosch CAN FD Specification V1.0

### (35) MTTCAN0TXESC — Tx Buffer Element Size Configuration

Configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0C8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TBDS[2:0] | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | RP | RP | RP |

**Table 37.104 MTTCAN0TXESC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 3 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 2 to 0 | TBDS[2:0] | Tx Buffer Data Field Size<br>000: 8 byte data field<br>001: 12 byte data field<br>010: 16 byte data field<br>011: 20 byte data field<br>100: 24 byte data field<br>101: 32 byte data field<br>110: 48 byte data field<br>111: 64 byte data field |

**NOTE**

In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size MTTCAN0TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as "$CC_H$" (padding bytes).

Bosch MCAN IP Specification Rev.3

### (36) MTTCAN0TXBRP — Tx Buffer Request Pending

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0CC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRP31 | TRP30 | TRP29 | TRP28 | TRP27 | TRP26 | TRP25 | TRP24 | TRP23 | TRP22 | TRP21 | TRP20 | TRP19 | TRP18 | TRP17 | TRP16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRP15 | TRP14 | TRP13 | TRP12 | TRP11 | TRP10 | TRP9 | TRP8 | TRP7 | TRP6 | TRP5 | TRP4 | TRP3 | TRP2 | TRP1 | TRP0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.105 MTTCAN0TXBRP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TRP[31:0] | Transmission Request Pending<br>Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register MTTCAN0TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register MTTCAN0TXBCR. MTTCAN0TXBRP bits are set only for those Tx Buffers configured via MTTCAN0TXBC. After a MTTCAN0TXBRP bit has been set, a Tx scan (see **Section 37.5.3.5, Tx Handling**) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID).<br>A cancellation request resets the corresponding transmission request pending bit of register MTTCAN0TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding MTTCAN0TXBRP bit has been reset.<br>After a cancellation has been requested, a finished cancellation is signalled via MTTCAN0TXBCF<br>• after successful transmission together with the corresponding MTTCAN0TXBTO bit<br>• when the transmission has not yet been started at the point of cancellation<br>• when the transmission has been aborted due to lost arbitration<br>• when an error occurred during frame transmission<br>In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding MTTCAN0TXBCF bit is set for all unsuccessful transmissions.<br>0: No transmission request pending<br>1: Transmission request pending<br>**NOTE**<br>MTTCAN0TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding MTTCAN0TXBRP bit is reset. |

Bosch CMOS Sensor Confidential

### (37) MTTCAN0TXBAR — Tx Buffer Add Request

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0D0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AR31 | AR30 | AR29 | AR28 | AR27 | AR26 | AR25 | AR24 | AR23 | AR22 | AR21 | AR20 | AR19 | AR18 | AR17 | AR16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AR15 | AR14 | AR13 | AR12 | AR11 | AR10 | AR9 | AR8 | AR7 | AR6 | AR5 | AR4 | AR3 | AR2 | AR1 | AR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.106 MTTCAN0TXBAR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | AR[31:0] | Add Request<br>Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to MTTCAN0TXBAR. MTTCAN0TXBAR bits are set only for those Tx Buffers configured via MTTCAN0TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.<br>0: No transmission request added<br>1: Transmission requested added |

**NOTE**

If an add request is applied for a Tx Buffer with pending transmission request (corresponding MTTCAN0TXBRP bit already set), this add request is ignored.

Preliminary Specification

### (38) MTTCAN0TXBCR — Tx Buffer Cancellation Request

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0D4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CR31 | CR30 | CR29 | CR28 | CR27 | CR26 | CR25 | CR24 | CR23 | CR22 | CR21 | CR20 | CR19 | CR18 | CR17 | CR16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CR15 | CR14 | CR13 | CR12 | CR11 | CR10 | CR9 | CR8 | CR7 | CR6 | CR5 | CR4 | CR3 | CR2 | CR1 | CR0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.107 MTTCAN0TXBCR Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CR[31:0] | Cancellation Request<br>Each Tx Buffer has its own Cancellation Request bit. Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to MTTCAN0TXBCR. MTTCAN0TXBCR bits are set only for those Tx Buffers configured via MTTCAN0TXBC. The bits remain set until the corresponding bit of MTTCAN0TXBRP is reset.<br>0: No cancellation pending<br>1: Cancellation pending |

Bosch CAN FD Specification Part

### (39) MTTCAN0TXBTO — Tx Buffer Transmission Occurred

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0D8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TO31 | TO30 | TO29 | TO28 | TO27 | TO26 | TO25 | TO24 | TO23 | TO22 | TO21 | TO20 | TO19 | TO18 | TO17 | TO16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TO15 | TO14 | TO13 | TO12 | TO11 | TO10 | TO9 | TO8 | TO7 | TO6 | TO5 | TO4 | TO3 | TO2 | TO1 | TO0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.108 MTTCAN0TXBTO Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TO[31:0] | Transmission Occurred<br>Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding MTTCAN0TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register MTTCAN0TXBAR.<br>0: No transmission occurred<br>1: Transmission occurred |

Bosch CAN FD Specification (Rev.1.0)

### (40) MTTCAN0TXBCF — Tx Buffer Cancellation Finished

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0DC_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF31 | CF30 | CF29 | CF28 | CF27 | CF26 | CF25 | CF24 | CF23 | CF22 | CF21 | CF20 | CF19 | CF18 | CF17 | CF16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CF15 | CF14 | CF13 | CF12 | CF11 | CF10 | CF9 | CF8 | CF7 | CF6 | CF5 | CF4 | CF3 | CF2 | CF1 | CF0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.109 MTTCAN0TXBCF Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CF[31:0] | Cancellation Finished<br>Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding MTTCAN0TXBRP bit is cleared after a cancellation was requested via MTTCAN0TXBCR. In case the corresponding MTTCAN0TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register MTTCAN0TXBAR.<br>0: No transmit buffer cancellation<br>1: Transmit buffer cancellation finished |

Bosch CAN FD Specification

### (41) MTTCAN0TXBTIE — Tx Buffer Transmission Interrupt Enable

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0E0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIE31 | TIE30 | TIE29 | TIE28 | TIE27 | TIE26 | TIE25 | TIE24 | TIE23 | TIE22 | TIE21 | TIE20 | TIE19 | TIE18 | TIE17 | TIE16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIE15 | TIE14 | TIE13 | TIE12 | TIE11 | TIE10 | TIE9 | TIE8 | TIE7 | TIE6 | TIE5 | TIE4 | TIE3 | TIE2 | TIE1 | TIE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.110 MTTCAN0TXBTIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | TIE[31:0] | Transmission Interrupt Enable<br>Each Tx Buffer has its own Transmission Interrupt Enable bit.<br>0: Transmission interrupt disabled<br>1: Transmission interrupt enable |

### (42) MTTCAN0TXBCIE — Tx Buffer Cancellation Finished Interrupt Enable

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0E4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIE31 | CFIE30 | CFIE29 | CFIE28 | CFIE27 | CFIE26 | CFIE25 | CFIE24 | CFIE23 | CFIE22 | CFIE21 | CFIE20 | CFIE19 | CFIE18 | CFIE17 | CFIE16 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFIE15 | CFIE14 | CFIE13 | CFIE12 | CFIE11 | CFIE10 | CFIE9 | CFIE8 | CFIE7 | CFIE6 | CFIE5 | CFIE4 | CFIE3 | CFIE2 | CFIE1 | CFIE0 |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 37.111 MTTCAN0TXBCIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 0 | CFIE[31:0] | Cancellation Finished Interrupt Enable<br>Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit.<br>0: Cancellation finished interrupt disabled<br>1: Cancellation finished interrupt enabled |

Bosch CAN FD Specification V1.0

### (43) MTTCAN0TXEFC — Tx Event FIFO Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0F0_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | EFWM[5:0] | | | | | | — | — | EFS[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | RP | RP | RP | RP | RP | RP | R | R | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EFSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.112 MTTCAN0TXEFC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31, 30 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 29 to 24 | EFWM[5:0] | Event FIFO Watermark<br>0: Watermark interrupt disabled<br>1 to 32: Level for Tx Event FIFO watermark interrupt (MTTCAN0IR.TEFW)<br>>32: Watermark interrupt disabled |
| 23, 22 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 21 to 16 | EFS[5:0] | Event FIFO Size<br>0: Tx Event FIFO disabled<br>1 to 32: Number of Tx Event FIFO elements<br>>32: Values greater than 32 are interpreted as 32<br>The Tx Event FIFO elements are indexed from 0 to EFS – 1 |
| 15 to 2 | EFSA[15:2] | Event FIFO Start Address<br>Start address of Tx Event FIFO in Message RAM (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

Boot CAN D Specification Ver.0

### (44) MTTCAN0TXEFS — Tx Event FIFO Status

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $0F4_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TEFL | EFF | — | — | — | EFPI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | EFGI[4:0] | | | | | — | — | EFFL[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.113 MTTCAN0TXEFS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. |
| 25 | TEFL | Tx Event FIFO Element Lost<br>This bit is a copy of interrupt flag MTTCAN0IR.TEFL. When MTTCAN0IR.TEFL is reset, this bit is also reset.<br>0: No Tx Event FIFO element lost<br>1: Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero. |
| 24 | EFF | Event FIFO Full<br>0: Tx Event FIFO not full<br>1: Tx Event FIFO full |
| 23 to 21 | Reserved | These bits are always read as 0. |
| 20 to 16 | EFPI[4:0] | Event FIFO Put Index<br>Tx Event FIFO write index pointer, range 0 to 31. |
| 15 to 13 | Reserved | These bits are always read as 0. |
| 12 to 8 | EFGI[4:0] | Event FIFO Get Index<br>Tx Event FIFO read index pointer, range 0 to 31. |
| 7, 6 | Reserved | These bits are always read as 0. |
| 5 to 0 | EFFL[5:0] | Event FIFO Fill Level<br>Number of elements stored in Tx Event FIFO, range 0 to 32. |

Bosch CAN FD Specification V1.0

### (45) MTTCAN0TXEFA — Tx Event FIFO Acknowledge

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $0F8_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | EFAI[4:0] | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

**Table 37.114 MTTCAN0TXEFA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 5 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 4 to 0 | EFAI[4:0] | Event FIFO Acknowledge Index<br>After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index MTTCAN0TXEFS.EFGI to EFAI + 1 and update the FIFO 0 Fill Level MTTCAN0TXEFS.EFFL. |

Bosch CAN FD Specification V1.0

### (46) MTTCAN0TTTMC — TT Trigger Memory Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $100_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | TME[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMSA[15:2] | | | | | | | | | | | | | | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | R |

**Table 37.115 MTTCAN0TTTMC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 22 to 16 | TME[6:0] | Trigger Memory Elements<br>0: No Trigger Memory<br>1 to 64: Number of Trigger Memory elements<br>>64: Values greater than 64 are interpreted as 64 |
| 15 to 2 | TMSA[15:2] | Trigger Memory Start Address<br>Start address of Trigger Memory in Message RAM (32-bit word address, see **Figure 37.4, Message RAM Configuration**). |
| 1, 0 | Reserved | These bits are always read as 0. When written, write the initial value. |

Bosch CAN FD Specification V1.0

### (47) MTTCAN0TTRMC — TT Reference Message Configuration

For details about handling of reference messages see **(1) Reference Message**

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $104_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMPS | XTD | — | RID[28:16] | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RID[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

**Table 37.116 MTTCAN0TTRMC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | RMPS | Reference Message Payload Select<br>Ignored in case of time slaves.<br>0: Reference message has no additional payload<br>1: The following elements are taken from Tx Buffer 0:<br>Message Marker MM, Event FIFO Control EFC, Data Length Code DLC, Data Bytes DB<br>(Level 1: bytes 2 to 8, Level 0,2: bytes 5 to 8) |
| 30 | XTD | Extended Identifier<br>0: 11-bit standard identifier<br>1: 29-bit extended identifier |
| 29 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 28 to 0 | RID[28:0] | Reference Identifier<br>Identifier transmitted with reference message and used for reference message filtering. Standard or extended reference identifier depending on bit XTD. A standard identifier has to be written to ID[28:18]. |

Bosch CAN Specification

### (48) MTTCAN0TTOCF — TT Operation Configuration

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $108_H$

**Value after reset:** $0001\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | EVTP | ECC | EGTF | AWL[7:0] | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EECS | IRTO[6:0] | | | | | | | LDSDL[2:0] | | | TM | GEN | — | OM[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | R | RP | RP |

**Table 37.117 MTTCAN0TTOCF Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 27 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 26 | EVTP | Event Trigger Polarity<br>0: Rising edge trigger<br>1: Falling edge trigger |
| 25 | ECC | Enable Clock Calibration<br>0: Automatic clock calibration in TTCAN Level 0, 2 is disabled<br>1: Automatic clock calibration in TTCAN Level 0, 2 is enabled |
| 24 | EGTF | Enable Global Time Filtering<br>0: Global time filtering in TTCAN Level 0, 2 is disabled<br>1: Global time filtering in TTCAN Level 0, 2 is enabled |
| 23 to 16 | AWL[7:0] | Application Watchdog Limit<br>The application watchdog can be disabled by programming AWL to 00H.<br>$00_H$ to $FF_H$: Maximum time after which the application has to serve the application watchdog.<br>The application watchdog is incremented once each 256 NTUs. |
| 15 | EECS | Enable External Clock Synchronization<br>If enabled, TUR configuration (MTTCAN0TURCF.NCL only) may be updated during TTCAN operation.<br>0: External clock synchronization in TTCAN Level 0, 2 disabled<br>1: External clock synchronization in TTCAN Level 0, 2 enabled |
| 14 to 8 | IRTO[6:0] | Initial Reference Trigger Offset<br>$00_H$ to $7F_H$: Positive offset, range from 0 to 127 |
| 7 to 5 | LDSDL[2:0] | LD of Synchronization Deviation Limit<br>The Synchronization Deviation Limit SDL is configured by its dual logarithm LDSDL with SDL = $2^{(LDSDL + 5)}$. It should not exceed the clock tolerance given by the CAN bit timing configuration.<br>$0_H$ to $7_H$: LD of Synchronization Deviation Limit (SDL ≤ 32 to 4096) |
| 4 | TM | Time Master<br>0: Time Master function disabled<br>1: Potential Time Master |

Bosch CAN FD Specification V1.0

Table 37.117 MTTCAN0TTOCF Register Contents (2/2)

| Bit Position | Bit Name | Function |
|---|---|---|
| 3 | GEN | Gap Enable<br>0: Strictly time-triggered operation<br>1: External event-synchronized time-triggered operation |
| 2 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 1, 0 | OM[1:0] | Operation Mode<br>00: Event-driven CAN communication, default<br>01: TTCAN level 1<br>10: TTCAN level 2<br>11: TTCAN level 0 |

Bosch CAN-FD Specification V1.0

### (49) MTTCAN0TTMLM — TT Matrix Limits

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $10C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ENTT[11:0] | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TXEW[3:0] | | | | CSS[1:0] | | CCM[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

**Table 37.118 MTTCAN0TTMLM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 28 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 27 to 16 | ENTT[11:0] | Expected Number of Tx Triggers<br>$000_H$ to $FFF_H$: Expected number of Tx Triggers in one Matrix Cycle |
| 15 to 12 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 11 to 8 | TXEW[3:0] | Tx Enable Window<br>$0_H$ to $F_H$: Length of Tx enable window, 1 to 16 NTU cycles |
| 7, 6 | CSS[1:0] | Cycle Start Synchronization<br>Enables sync pulse output at pin m_ttcan_soc (MTTCAN0SOC).<br>00: No sync pulse<br>01: Sync pulse at start of basic cycle<br>10: Sync pulse at start of matrix cycle<br>11: Reserved |
| 5 to 0 | CCM[5:0] | Cycle Count Max<br>$00_H$: 1 Basic Cycle per Matrix Cycle<br>$01_H$: 2 Basic Cycles per Matrix Cycle<br>$03_H$: 4 Basic Cycles per Matrix Cycle<br>$07_H$: 8 Basic Cycles per Matrix Cycle<br>$0F_H$: 16 Basic Cycles per Matrix Cycle<br>$1F_H$: 32 Basic Cycles per Matrix Cycle<br>$3F_H$: 64 Basic Cycles per Matrix Cycle<br>others: Reserved |

Bosch CAN FD Specification V1.0

**NOTE**

*ISO 11898-4, Section 5.2.1 requires,* that only the listed cycle count values are configured. Other values are possible but may lead to inconsistent matrix cycles.

### (50) MTTCAN0TURCF — TUR Configuration

The length of the NTU is given by: NTU = CAN Clock Period • NC/DC

NC is an 18-bit value. Its high part, NCH[17:16] is hard wired to 0b01. Therefore the range of NC is $10000_H$ to $1FFFF_H$. The value configured by NCL is the initial value for MTTCAN0TURNA.NAV[15:0]. DC is set to $1000_H$ by hardware reset and it may not be written to $0000_H$.

Level 1: NC ≥ 4 • DC and NTU = CAN bit time
Level 0, 2: NC ≥ 8 • DC

The actual value of TUR may be changed by the clock drift compensation function of TTCAN Level 0 and Level 2 in order to adjust the node's local view of the NTU to the time master's view of the NTU. DC will not be changed by the automatic drift compensation, MTTCAN0TURNA.NAV may be adjusted around NC in the range of the Synchronization Deviation Limit given by MTTCAN0TTOCF.LDSDL. NC and DC should be programmed to the largest suitable values in order to allow the best computational accuracy for the drift compensation process.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $110_H$

**Value after reset:** $1000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ELT | — | DC[13:0] | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | R | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NCL[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP | RP |

Bosch CAN-FD Specification V1.0

**Table 37.119 MTTCAN0TURCF Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | ELT | Enable Local Time<br>0: Local time is stopped, default<br>1: Local time is enabled<br>**NOTE**<br>Local time is started by setting ELT. It remains active until ELT is reset or until the next hardware reset. MTTCAN0TURCF.DC is locked when MTTCAN0TURCF.ELT = '1'. If ELT is written to '0', the readable value will stay at '1' until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register remains locked. |
| 30 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 29 to 16 | DC[13:0] | Denominator Configuration<br>$0000_H$: Illegal value<br>$0001_H$ to $3FFF_H$: Denominator Configuration |

**Table 37.119 MTTCAN0TURCF Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 15 to 0 | NCL[15:0] | Numerator Configuration Low<br>Write access to the TUR Numerator Configuration Low is only possible during configuration with MTTCAN0TURCF.ELT = '0' or if MTTCAN0TTOCF.EECS (external clock synchronization enabled) is set. When a new value for NCL is written outside TT Configuration Mode, the new value takes effect when MTTCAN0TTOST.WECS is cleared to '0'. NCL is locked MTTCAN0TTOST.WECS is '1'.<br>$0000_H$ to $FFFF_H$: Numerator Configuration Low<br>**NOTE**<br>If NC < 7 • DC in TTCAN Level 1, then it is required that subsequent time marks in the Trigger Memory must differ by at least 2 NTU. |

Bosch CAN-FD Specification V1.0

### (51) MTTCAN0TTOCN — TT Operation Control

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $114_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LCKC | — | ESCN | NIG | TMG | FGP | GCS | TTIE | TMC[1:0] | | RTIE | SWS[1:0] | | SWP | ECS | SGT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW |

**Table 37.120 MTTCAN0TTOCN Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 26 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 15 | LCKC | TT Operation Control Register Locked<br>Set by a write access to register MTTCAN0TTOCN. Reset when the updated configuration has been synchronized into the CAN clock domain.<br>0: Write access to MTTCAN0TTOCN enabled<br>1: Write access to MTTCAN0TTOCN locked |
| 14 | Reserved | This bit is always read as 0. When written, write the initial value. |
| 13 | ESCN | External Synchronization Control<br>If enabled the M_TTCAN synchronizes its cycle time phase to an external event signalled by a rising edge at pin m_ttcan_evt (MTTCAN0EVT)(see **Section 37.6.4.10, Synchronization to external Time Schedule**).<br>0: External synchronization disabled<br>1: External synchronization enabled |
| 12 | NIG | Next is Gap<br>This bit can only be set when the M_TTCAN is the actual Time Master and when it is configured for external event-synchronized time-triggered operation (MTTCAN0TTOCF.GEN = '1')<br>0: No action, reset by reception of any reference message<br>1: Transmit next reference message with Next_is_Gap = '1' |
| 11 | TMG | Time Mark Gap<br>0: Reset by each reference message<br>1: Next reference message started when Register Time Mark interrupt MTTCAN0TTIR.RTMI is activated |
| 10 | FGP | Finish Gap<br>Set by the CPU, reset by each reference message<br>0: No reference message requested<br>1: Application requested start of reference message |
| 9 | GCS | Gap Control Select<br>0: Gap control independent from m_ttcan_evt (MTTCAN0EVT)<br>1: Gap control by input pin m_ttcan_evt (MTTCAN0EVT) |
| 8 | TTIE | Trigger Time Mark Interrupt Pulse Enable<br>External time mark events are configured by trigger memory element TMEX (see **(7) Trigger Memory Element**). A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the M_TTCAN is in synchronization state In_Schedule or In_Gap.<br>0: Trigger Time Mark Interrupt output m_ttcan_tmp (MTTCAN0TMP) disabled<br>1: Trigger Time Mark Interrupt output m_ttcan_tmp (MTTCAN0TMP) enabled |

Bosch CAN IP Specification

**Table 37.120 MTTCAN0TTOCN Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 7, 6 | TMC[1:0] | Register Time Mark Compare<br>00: No Register Time Mark Interrupt generated<br>01: Register Time Mark Interrupt if Time Mark = cycle time<br>10: Register Time Mark Interrupt if Time Mark = local time<br>11: Register Time Mark Interrupt if Time Mark = global time<br>**NOTE**<br>When changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = “00”, then reconfigure MTTCAN0TTTMK, and finally set TMC to the intended time reference. |
| 5 | RTIE | Register Time Mark Interrupt Pulse Enable<br>Register time mark interrupts are configured by register MTTCAN0TTTMK. A register time mark interrupt pulse with the length of one m_ttcan_clk period is generated when the time referenced by MTTCAN0TTOCN.TMC (cycle, local, or global) equals MTTCAN0TTTMK.TM, independent of the synchronization state.<br>0: Register Time Mark Interrupt output m_ttcan_rtp (MTTCAN0RTP) disabled<br>1: Register Time Mark Interrupt output m_ttcan_rtp (MTTCAN0RTP) enabled |
| 4, 3 | SWS[1:0] | Stop Watch Source<br>00: Stop Watch disabled<br>01: Actual value of cycle time is copied to MTTCAN0TTCPT.SWV<br>10: Actual value of local time is copied to MTTCAN0TTCPT.SWV<br>11: Actual value of global time is copied to MTTCAN0TTCPT.SWV |
| 2 | SWP | Stop Watch Polarity<br>0: Rising edge trigger<br>1: Falling edge trigger |
| 1 | ECS | External Clock Synchronization<br>Writing a ‘1’ to ECS sets MTTCAN0TTOST.WECS if the node is the actual Time Master. ECS is reset after one Host clock period. The external clock synchronization takes effect at the start of the next basic cycle. |
| 0 | SGT | Set Global time<br>Writing a ‘1’ to SGT sets MTTCAN0TTOST.WGDT if the node is the actual Time Master. SGT is reset after one Host clock period. The global time preset takes effect when the node transmits the next reference message with the Master_Ref_Mark modified by the preset value written to MTTCAN0TTGTP. |

Bosch CAN FD Specification V1.0

### (52) MTTCAN0TTGTP — TT Global Time Preset

If MTTCAN0TTOST.WGDT is set, the next reference message will be transmitted with the Master_Ref_Mark modified by the preset value and with Disc_Bit = '1', presetting the global time in all nodes simultaneously.

TP is reset to $0000_H$ each time a reference message with Disc_Bit = '1' becomes valid or if the node is not the current Time Master. TP is locked while MTTCAN0TTOST.WGTD = '1' after setting MTTCAN0TTOCN.SGT until the reference message with Disc_Bit = '1' becomes valid or until the node is no longer the current Time Master.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $118_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CTP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

Bosch CAN-FD Specification V1.0

**Table 37.121 MTTCAN0TTGTP Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | CTP[15:0] | Cycle Time Target Phase<br>CTP is write-protected while MTTCAN0TTOCN.ESCN or MTTCAN0TTOST.SPL are set (see **Section 37.6.4.10, Synchronization to external Time Schedule**).<br>$0000_H$ to $FFFF_H$: Defines target value of cycle time when a rising edge of m_ttcan_evt (MTTCAN0EVT) is expected |
| 15 to 0 | TP[15:0] | Time Preset<br>TP is write-protected while MTTCAN0TTOST.WGTD is set.<br>$0000_H$ to $7FFF_H$: Next Master Reference Mark = Master Reference Mark + TP<br>$8000_H$: reserved<br>$8001_H$ to $FFFF_H$: Next Master Reference Mark = Master Reference Mark – ($10000_H$ – TP) |

### (53) MTTCAN0TTTMK — TT Time Mark

A time mark interrupt (MTTCAN0TTIR.RTMI = '1') is generated when the time base indicated by MTTCAN0TTOCN.TMC (cycle time, local time, or global time) has the same value as TM.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $11C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LCKM | — | — | — | — | — | — | — | — | TICC[6:0] | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R/W | RW | R/W | RW | R/W | RW | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

**Table 37.122 MTTCAN0TTTMK Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | LCKM | TT Time Mark Register Locked<br>Always set by a write access to registers MTTCAN0TTOCN. Set by write access to register MTTCAN0TTTMK when MTTCAN0TTOCN.TMC ≠ "00".<br>Reset when the registers have been synchronized into the CAN clock domain.<br>0: Write access to MTTCAN0TTTMK enabled<br>1: Write access to MTTCAN0TTTMK locked |
| 30 to 23 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 22 to 16 | TICC[6:0] | Time Mark Cycle Code<br>Cycle count for which the time mark is valid.<br>000000x: valid for all cycles<br>000001c: valid every second cycle at cycle count mod2 = c<br>00001cc: valid every fourth cycle at cycle count mod4 = cc<br>0001ccc: valid every eighth cycle at cycle count mod8 = ccc<br>001cccc: valid every sixteenth cycle at cycle count mod16 = cccc<br>01ccccc: valid every thirty-second cycle at cycle count mod32 = ccccc<br>1cccccc: valid every sixty-fourth cycle at cycle count mod64 = cccccc |
| 15 to 0 | TM[15:0] | Time Mark<br>$0000_H$ to $FFFF_H$: Time Mark |

**NOTE**

When using byte access to register MTTCAN0TTTMK it is recommended to first disable the time mark compare function (MTTCAN0TTOCN.TMC = "00") to avoid compares on inconsistent register values.

Specification Sheet

### (54) MTTCAN0TTIR — TT Interrupt Register

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $120_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CER | AW | WT |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | RW | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IWT | ELC | SE2 | SE1 | TXO | TXU | GTE | GTD | GTW | SWE | TTMI | RTMI | SOG | CSM | SMC | SBC |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

**Table 37.123 MTTCAN0TTIR Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 18 | CER | Configuration Error<br>Trigger out of order.<br>0: No error found in trigger list<br>1: Error found in trigger list |
| 17 | AW | Application Watchdog<br>0: Application watchdog served in time<br>1: Application watchdog not served in time |
| 16 | WT | Watch Trigger<br>0: No missing reference message<br>1: Missing reference message (Level 0: cycle time FF00H) |
| 15 | IWT | Initialization Watch Trigger<br>The initialization is restarted by resetting IWT.<br>0: No missing reference message during system startup<br>1: No system startup due to missing reference message |
| 14 | ELC | Error Level Changed<br>Not set when error level changed during initialization.<br>0: No change in error level<br>1: Error level changed |
| 13 | SE2 | Scheduling Error 2<br>0: No scheduling error 2<br>1: Scheduling error 2 occurred |
| 12 | SE1 | Scheduling Error 1<br>0: No scheduling error 1<br>1: Scheduling error 1 occurred |
| 11 | TXO | Tx Count Overflow<br>0: Number of Tx Trigger as expected<br>1: More Tx trigger than expected in one matrix cycle |
| 10 | TXU | Tx Count Underflow<br>0: Number of Tx Trigger as expected<br>1: Less Tx trigger than expected in one matrix cycle |

Bosch CAN FD Specification V1.0

**Table 37.123 MTTCAN0TTIR Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 9 | GTE | Global Time Error<br>Synchronization deviation SD exceeds limit specified by MTTCAN0TTOCF.LDSDL, TTCAN Level 0, 2 only.<br>0: Synchronization deviation within limit<br>1: Synchronization deviation exceeded limit |
| 8 | GTD | Global Time Discontinuity<br>0: No discontinuity of global time<br>1: Discontinuity of global time |
| 7 | GTW | Global Time Wrap<br>0: No global time wrap occurred<br>1: Global time wrap from $FFFF_H$ to $0000_H$ occurred |
| 6 | SWE | Stop Watch Event<br>0: No rising/falling edge at stop watch trigger pin m_ttcan_swt (MTTCAN0SWT) detected<br>1: Rising/falling edge at stop watch trigger pin m_ttcan_swt (MTTCAN0SWT) detected |
| 5 | TTMI | Trigger Time Mark Event Internal<br>Internal time mark events are configured by trigger memory element TMIN (see **(7) Trigger Memory Element**). Set when the trigger memory element becomes active, and the M_TTCAN is in synchronization state In_Gap or In_Schedule.<br>0: Time mark not reached<br>1: Time mark reached (Level 0: cycle time MTTCAN0TTOCF.IRTO • $200_H$) |
| 4 | RTMI | Register Time Mark Interrupt<br>Set when time referenced by MTTCAN0TTOCN.TMC (cycle, local, or global) equals MTTCAN0TTTMK.TM, independent of the synchronization state.<br>0: Time mark not reached<br>1: Time mark reached |
| 3 | SOG | Start of Gap<br>0: No reference message seen with Next_is_Gap bit set<br>1: Reference message with Next_is_Gap bit set becomes valid |
| 2 | CSM | Change of Synchronization Mode<br>0: No change in master to slave relation or schedule synchronization<br>1: Master to slave relation or schedule synchronization changed, also set when MTTCAN0TTOST.SPL is reset |
| 1 | SMC | Start of Matrix Cycle<br>0: No Matrix Cycle started since bit has been reset<br>1: Matrix Cycle started |
| 0 | SBC | Start of Basic Cycle<br>0: No Basic Cycle started since bit has been reset<br>1: Basic Cycle started |

Bosch CAN FD Specification V1.0

### (55) MTTCAN0TTIE — TT Interrupt Enable

The settings in the TT Interrupt Enable register determine which status changes in the TT Interrupt Register will result in an interrupt.

0: TT interrupt disabled

1: TT interrupt enabled

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $124_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CERE | AWE | WTE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | RW | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IWTE | ELCE | SE2E | SE1E | TXOE | TXUE | GTEE | GTDE | GTWE | SWEE | TTMIE | RTMIE | SOGE | CSME | SMCE | SBCE |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

Bosch CAN-FD Specification V1.0

**Table 37.124 MTTCAN0TTIE Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 18 | CERE | Configuration Error Interrupt Enable |
| 17 | AWE | Application Watchdog Interrupt Enable |
| 16 | WTE | Watch Trigger Interrupt Enable |
| 15 | IWTE | Initialization Watch Trigger Interrupt Enable |
| 14 | ELCE | Change Error Level Interrupt Enable |
| 13 | SE2E | Scheduling Error 2 Interrupt Enable |
| 12 | SE1E | Scheduling Error 1 Interrupt Enable |
| 11 | TXOE | Tx Count Overflow Interrupt Enable |
| 10 | TXUE | Tx Count Underflow Interrupt Enable |
| 9 | GTEE | Global Time Error Interrupt Enable |
| 8 | GTDE | Global Time Discontinuity Interrupt Enable |
| 7 | GTWE | Global Time Wrap Interrupt Enable |
| 6 | SWEE | Stop Watch Event Interrupt Enable |
| 5 | TTMIE | Trigger Time Mark Event Internal Enable |
| 4 | RTMIE | Register Time Mark Interrupt Enable |
| 3 | SOGE | Start of Gap Interrupt Enable |
| 2 | CSME | Change of Synchronization Mode Interrupt Enable |
| 1 | SMCE | Start of Matrix Cycle Interrupt Enable |
| 0 | SBCE | Start of Basic Cycle Interrupt Enable |

### (56) MTTCAN0TTILS — TT Interrupt Line Select

The TT Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the TT Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via MTTCAN0ILE.EINT0 and MTTCAN0ILE.EINT1.

0: TT interrupt assigned to interrupt line m_ttcan_int0 (INTMTTCANI0)

1: TT interrupt assigned to interrupt line m_ttcan_int1 (INTMTTCANI1)

**Access:** This register can be read/written in 32-bit units.

**Address:** <MTTCAN0_base> + $128_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CERL | AWL | WTL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | RW | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IWTL | ELCL | SE2L | SE1L | TXOL | TXUL | GTEL | GTDL | GTWL | SWEL | TTMIL | RTMIL | SOGL | CSML | SMCL | SBCL |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W | RW | R/W |

Bosch CAN-FD Specification V1.0

**Table 37.125 MTTCAN0TTILS Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 19 | Reserved | These bits are always read as 0. When written, write the initial value. |
| 18 | CERL | Configuration Error Interrupt Line |
| 17 | AWL | Application Watchdog Interrupt Line |
| 16 | WTL | Watch Trigger Interrupt Line |
| 15 | IWTL | Initialization Watch Trigger Interrupt Line |
| 14 | ELCL | Change Error Level Interrupt Line |
| 13 | SE2L | Scheduling Error 2 Interrupt Line |
| 12 | SE1L | Scheduling Error 1 Interrupt Line |
| 11 | TXOL | Tx Count Overflow Interrupt Line |
| 10 | TXUL | Tx Count Underflow Interrupt Line |
| 9 | GTEL | Global Time Error Interrupt Line |
| 8 | GTDL | Global Time Discontinuity Interrupt Line |
| 7 | GTWL | Global Time Wrap Interrupt Line |
| 6 | SWEL | Stop Watch Event Interrupt Line |
| 5 | TTMIL | Trigger Time Mark Event Internal Line |
| 4 | RTMIL | Register Time Mark Interrupt Line |
| 3 | SOGL | Start of Gap Interrupt Line |
| 2 | CSML | Change of Synchronization Mode Interrupt Line |
| 1 | SMCL | Start of Matrix Cycle Interrupt Line |
| 0 | SBCL | Start of Basic Cycle Interrupt Line |

### (57) MTTCAN0TTOST — TT Operation Status

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $12C_H$

**Value after reset:** $0000\ 0080_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPL | WECS | AWE | WFE | GSI | TMP[2:0] | | | GFI | WGTD | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RTO[7:0] | | | | | | | | QCS | QGTP | SYS[1:0] | | MS[1:0] | | EL[1:0] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.126 MTTCAN0TTOST Register Contents (1/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 | SPL | Schedule Phase Lock<br>The bit is valid only when external synchronization is enabled (MTTCAN0TTOCN.ESCN = '1'). In this case it signals that the difference between cycle time configured by MTTCAN0TTGTP.CTP and the cycle time at the rising edge at pin m_ttcan_evt (MTTCAN0EVT) is less or equal 9 NTU (see **Section 37.6.4.10, Synchronization to external Time Schedule**).<br>0: Phase outside range<br>1: Phase inside range |
| 30 | WECS | Wait for External Clock Synchronization<br>0: No external clock synchronization pending<br>1: Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle. |
| 29 | AWE | Application Watchdog Event<br>The application watchdog is served by reading MTTCAN0TTOST. When the watchdog is not served in time, bit AWE is set, all TTCAN communication is stopped, and the M_TTCAN is set into Bus Monitoring Mode.<br>0: Application Watchdog served in time<br>1: Failed to serve Application Watchdog in time |
| 28 | WFE | Wait for Event<br>0: No Gap announced, reset by a reference message with Next_is_Gap = '0'<br>1: Reference message with Next_is_Gap = '1' received |
| 27 | GSI | Gap Started Indicator<br>0: No Gap in schedule, reset by each reference message and for all time slaves<br>1: Gap time after Basic Cycle has started |
| 26 to 24 | TMP[2:0] | Time Master Priority<br>$0_H$ to $7_H$: Priority of actual Time Master |
| 23 | GFI | Gap Finished Indicator<br>Set when the CPU writes MTTCAN0TTOCN.FGP, or by a time mark interrupt if TMG = '1', or via input pin m_ttcan_evt (MTTCAN0EVT) if MTTCAN0TTOCN.GCS = '1'. Not set by Ref_Trigger_Gap or when Gap is finished by another node sending a reference message.<br>0: Reset at the end of each reference message<br>1: Gap finished by M_TTCAN |
| 22 | WGTD | Wait for Global Time Discontinuity<br>0: No global time preset pending<br>1: Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a reference message with Disc_Bit = '1' or after it received a reference message. |

Bosch CAN FD Specification 1.0

**Table 37.126 MTTCAN0TTOST Register Contents (2/2)**

| Bit Position | Bit Name | Function |
|---|---|---|
| 21 to 16 | Reserved | These bits are always read as 0. |
| 15 to 8 | RTO[7:0] | Reference Trigger Offset<br>The Reference Trigger Offset value is a signed integer with a range from -127 ($81_H$) to 127 ($7F_H$). There is no notification when the lower limit of -127 is reached. In case the M_TTCAN becomes Time Master (MS[1:0] = “11”), the reset of RTO is delayed due to synchronization between Host and CAN clock domain. For time slaves the value configured by MTTCAN0TTOCF.IRTO is read.<br>$00_H$ to $FF_H$: Actual Reference Trigger offset value |
| 7 | QCS | Quality of Clock Speed<br>Only relevant in TTCAN Level 0 and Level 2, otherwise fixed to ‘1’.<br>0: Local clock speed not synchronized to Time Master clock speed<br>1: Synchronization Deviation ≤ SDL |
| 6 | QGTP | Quality of Global Time Phase<br>Only relevant in TTCAN Level 0 and Level 2, otherwise fixed to ‘0’.<br>0: Global time not valid<br>1: Global time in phase with Time Master |
| 5, 4 | SYS[1:0] | Synchronization State<br>00: Out of Synchronization<br>01: Synchronizing to TTCAN communication<br>10: Schedule suspended by Gap (In_Gap)<br>11: Synchronized to schedule (In_Schedule) |
| 3, 2 | MS[1:0] | Master State<br>00: Master_Off, no master properties relevant<br>01: Operating as Time Slave<br>10: Operating as Backup Time Master<br>11: Operating as current Time Master |
| 1, 0 | EL[1:0] | Error Level<br>00: Severity 0 - No Error<br>01: Severity 1 - Warning<br>10: Severity 2 - Error<br>11: Severity 3 - Severe Error |

Bosch CAN-FD Specification V1.0

### (58) MTTCAN0TURNA — TUR Numerator Actual

There is no drift compensation in TTCAN Level 1 (NAV = NC). In TTCAN Level 0 and Level 2, the drift between the node's local clock and the time master's local clock is calculated. The drift is compensated when the Synchronization Deviation (difference between NC and the calculated NAV) is not more than 2 (MTTCAN0TTOCF.LDSDL + 5). With MTTCAN0TTOCF.LDSDL ≤ 7, this results in a maximum range for NAV of $(NC - 1000_H) \leq NAV \leq (NC + 1000_H)$.

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $130_H$

**Value after reset:** 0000 $0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | NAV[17:16] | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NAV[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Boot CAN-FD Specification V1.0

**Table 37.127 MTTCAN0TURNA Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 18 | Reserved | These bits are always read as 0. |
| 17 to 0 | NAV[17:0] | Numerator Actual Value<br>≤ $0EFFF_H$: Illegal value<br>$0F000_H$ to $20FFF_H$: Actual numerator value<br>≥ $21000_H$: Illegal value |

### (59) MTTCAN0TTLGT — TT Local & Global Time

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $134_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.128 MTTCAN0TTLGT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | GT[15:0] | Global Time<br>Non-fractional part of the sum of the node's local time and its local offset (see **Section 37.6.4.5, Local Time, Cycle Time, Global Time, and External Clock Synchronization**).<br>$0000_H$ to $FFFF_H$: Global time value of TTCAN network |
| 15 to 0 | LT[15:0] | Local Time<br>Non-fractional part of local time, incremented once each local NTU (see **Section 37.6.4.5, Local Time, Cycle Time, Global Time, and External Clock Synchronization**).<br>$0000_H$ to $FFFF_H$: Local time value of TTCAN node |

Bosch CAN-FD Specification V1.0

### (60) MTTCAN0TTCTC — TT Cycle Time & Count

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $138_H$

**Value after reset:** $003F\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CC[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CT[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.129 MTTCAN0TTCTC Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 22 | Reserved | These bits are always read as 0. |
| 21 to 16 | CC[5:0] | Cycle Count<br>$00_H$ to $3F_H$: Number of actual Basic Cycle in the System Matrix |
| 15 to 0 | CT[15:0] | Cycle Time<br>Non-fractional part of the difference of the node's local time and Ref_Mark (see **Section 37.6.4.5, Local Time, Cycle Time, Global Time, and External Clock Synchronization**).<br>$0000_H$ to $FFFF_H$: Cycle time value of TTCAN Basic Cycle |

Bosch CAN FD Specification V1.0

### (61) MTTCAN0TTCPT — TT Capture Time

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $13C_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SWV[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CCV[5:0] | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.130 MTTCAN0TTCPT Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | SWV[15:0] | Stop Watch Value<br>On a rising/falling edge (as configured via MTTCAN0TTOCN.SWP) at the Stop Watch Trigger pin m_ttcan_swt (MTTCAN0SWT) , when MTTCAN0TTOCN.SWS is ≠ “00” and MTTCAN0TTIR.SWE is ‘0’, the actual time value as selected by MTTCAN0TTOCN.SWS (cycle, local, global) is copied to SWV and MTTCAN0TTIR.SWE will be set to ‘1’. Capturing of the next stop watch value is enabled by resetting MTTCAN0TTIR.SWE.<br>$0000_H$ to $FFFF_H$: Captured Stop Watch value |
| 15 to 6 | Reserved | These bits are always read as 0. |
| 5 to 0 | CCV[5:0] | Cycle Count Value<br>Cycle count value captured together with SWV.<br>$00_H$ to $3F_H$: Captured cycle count value |

Bosch CAN FD Specification

### (62) MTTCAN0TTCSM — TT Cycle Sync Mark

**Access:** This register can be read in 32-bit units.

**Address:** <MTTCAN0_base> + $140_H$

**Value after reset:** $0000\ 0000_H$

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSM[15:0] | | | | | | | | | | | | | | | |
| Value after reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

**Table 37.131 MTTCAN0TTCSM Register Contents**

| Bit Position | Bit Name | Function |
|---|---|---|
| 31 to 16 | Reserved | These bits are always read as 0. |
| 15 to 0 | CSM[15:0] | Cycle Sync Mark<br>The Cycle Sync Mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid.<br>$0000_H$ to $FFFF_H$: Captured cycle time |

Bosch CAN FD Specification V1.0

### 37.6.2.4 Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration a single-ported Message RAM is connected to the M_TTCAN module.

#### (1) Message RAM Configuration

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_TTCAN module can be configured to allocate up to 4480 words in the Message RAM. It is not necessary to configure each of the sections listed in **Figure 37.15, Message RAM Configuration**, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via MTTCAN0RXESC.F0DS, MTTCAN0RXESC.F1DS, MTTCAN0RXESC.RBDS, and MTTCAN0TXESC.TBDS.

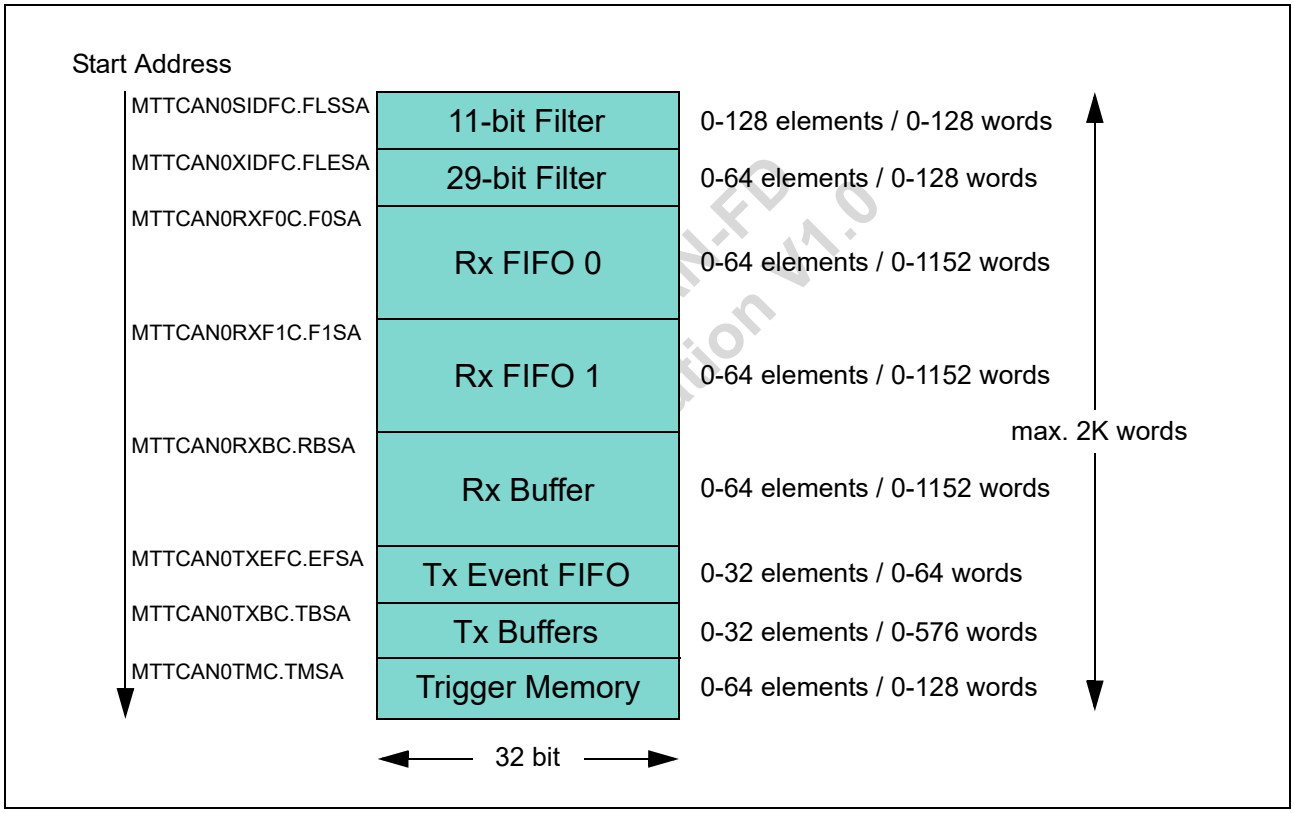



**Figure 37.15 Message RAM Configuration**

When the M_TTCAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

**NOTE**

The M_TTCAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

### (2) Rx Buffer and FIFO Element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in **Table 37.132** below. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register MTTCAN0RXESC.

**Table 37.132 Rx Buffer and FIFO Element**

| | 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|---|
| R0 | ESI, XTD, RTR, ID[28:0] | | | |
| R1 | ANMF, FIDX[6:0] | res, EDL, BRS, DLC[3:0] | RXTS[15:0] | |
| R2 | DB3[7:0] | DB2[7:0] | DB1[7:0] | DB0[7:0] |
| R3 | DB7[7:0] | DB6[7:0] | DB5[7:0] | DB4[7:0] |
| ... | ... | ... | ... | ... |
| Rn | DBm[7:0] | DBm-1[7:0] | DBm-2[7:0] | DBm-3[7:0] |

Bosch CAN-FD Specification V1.0

**R0 Bit 31 ESI: Error State Indicator**

0: Transmitting node is error active
1: Transmitting node is error passive

**R0 Bit 30 XTD: Extended Identifier**

Signals to the Host whether the received frame has a standard or extended identifier.

0: 11-bit standard identifier
1: 29-bit extended identifier

**R0 Bit 29 RTR: Remote Transmission Request**

Signals to the Host whether the received frame is a data frame or a remote frame.

0:Received frame is a data frame

1:Received frame is a remote frame

**NOTE**

There are no remote frames in CAN FD format. In case a CAN FD frame was received (EDL = 1'), bit RTR reflects the state of the reserved bit r1.

**R0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

**R1 Bit 31 ANMF: Accepted Non-matching Frame**

Acceptance of non-matching frames may be enabled via MTTCAN0GFC.ANFS and MTTCAN0GFC.ANFE.

0:Received frame matching filter index FIDX

1:Received frame did not match any Rx filter element

**R1 Bits 30:24 FIDX[6:0]:Filter Index**

0 to 127:Index of matching Rx acceptance filter element (invalid if ANMF = '1').
Range is 0 to MTTCAN0SIDFC.LSS - 1 resp. MTTCAN0XIDFC.LSE - 1.

**R1 Bit 21 EDL: Extended Data Length**

0:Standard frame format

1:CAN FD frame format (new DLC-coding and CRC)

**R1 Bit 20 BRS: Bit Rate Switch**

0:Frame received without bit rate switching

1:Frame received with bit rate switching

**R1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: received frame has 0 to 8 data bytes

9 to 15: CAN: received frame has 8 data bytes

9 to 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes

**R1 Bits 15:0 RXTS[15:0]:Rx Timestamp**

Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler MTTCAN0TSCC.TCP.

Bosch CAN-FD Specification V1.0

**R2 Bits 31:24 DB3[7:0]:Data Byte 3**

**R2 Bits 23:16 DB2[7:0]:Data Byte 2**

**R2 Bits 15:8 DB1[7:0]:Data Byte 1**

**R2 Bits 7:0 DB0[7:0]:Data Byte 0**

**R3 Bits 31:24 DB7[7:0]:Data Byte 7**

**R3 Bits 23:16 DB6[7:0]:Data Byte 6**

**R3 Bits 15:8 DB5[7:0]:Data Byte 5**

**R3 Bits 7:0 DB4[7:0]:Data Byte 4**

...

**Rn Bits 31:24 DBm[7:0]:Data Byte m**

**Rn Bits 23:16 DBm-1[7:0]:Data Byte m-1**

**Rn Bits 15:8 DBm-2[7:0]:Data Byte m-2**

**Rn Bits 7:0 DBm-3[7:0]:Data Byte m-3**

**NOTE**

Depending on the configuration of the element size (MCTTAN0RXESC), between two and sixteen 32-bit words (Rn = 3 to 17) are used for storage of a CAN message's data field.

Bosch CAN-FD Specification V1.0

### (3) Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration MTTCAN0TXBC.TFQS and MTTCAN0TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register MTTCAN0TXESC.

**Table 37.133 Tx Buffer Element**

| | 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|---|
| T0 | res, XTD, RTR, ID[28:0] | | | |
| T1 | MM[7:0] | EFC, res, DLC[3:0] | res | |
| T2 | DB3[7:0] | DB2[7:0] | DB1[7:0] | DB0[7:0] |
| T3 | DB7[7:0] | DB6[7:0] | DB5[7:0] | DB4[7:0] |
| ... | ... | ... | ... | ... |
| Tn | DBm[7:0] | DBm-1[7:0] | DBm-2[7:0] | DBm-3[7:0] |

Bosch CAN-FD Specification V1.0

**T0 Bit 30 XTD: Extended Identifier**

0: 11-bit standard identifier

1: 29-bit extended identifier

**T0 Bit 29 RTR: Remote Transmission Request**

0: Transmit data frame

1: Transmit remote frame

**NOTE**

When RTR = 1, the M_TTCAN transmits a remote frame according to ISO11898-1, even if MTTCAN0CCCR.CME enables the transmission in CAN FD format.

**T0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

**T1 Bits 31:24 MM[7:0]: Message Marker**

Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.

**T1 Bit 23 EFC: Event FIFO Control**

0: Don’t store Tx events

1: Store Tx events

**T1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: transmit frame has 0-8 data bytes

9 to 15: CAN: transmit frame has 8 data bytes

9 to 15: CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes

**T2 Bits 31:24DB3[7:0]:Data Byte 3**

**T2 Bits 23:16 DB2[7:0]: Data Byte 2**

**T2 Bits 15:8 DB1[7:0]: Data Byte 1**

**T2 Bits 7:0 DB0[7:0]: Data Byte 0**

**T3 Bits 31:24 DB7[7:0]: Data Byte 7**

**T3 Bits 23:16 DB6[7:0]: Data Byte 6**

**T3 Bits 15:8 DB5[7:0]: Data Byte 5**

**T3 Bits 7:0 DB4[7:0]: Data Byte 4**

**...**

**Tn Bits 31:24 DBm[7:0]: Data Byte m**

**Tn Bits 23:16 DBm-1[7:0]:Data Byte m-1**

**Tn Bits 15:8 DBm-2[7:0]:Data Byte m-2**

**Tn Bits 7:0 DBm-3[7:0]:Data Byte m-3**

Bosch CAN-FD Specification V1.0

**NOTE**

Depending on the configuration of the element size (MTTCAN0TXESC), between two and sixteen 32-bit words (Tn = 3 to 17) are used for storage of a CAN message’s data field.

### (4) Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register MTTCAN0TXEFS.

**Table 37.134 Tx Event FIFO Element**

| | 31 | 30 | 29 | 28 | 24 | 23 | 22 | 21 | 20 | 19 | 16 | 15 | 8 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E0 | ESI | XTD | RTR | ID[28:0] | | | | | | | | | | | |
| E1 | MM[7:0] | | | | | ET[1:0] | | EDL | BRS | DLC[3:0] | | TXTS[15:0] | | | |

**E0 Bit 31 ESI: Error State Indicator**

0: Transmitting node is error active

1: Transmitting node is error passive

**E0 Bit 30 XTD: Extended Identifier**

0: 11-bit standard identifier

1: 29-bit extended identifier

**E0 Bit 29 RTR: Remote Transmission Request**

0: Data frame transmitted

1: Remote frame transmitted

**E0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

**E1 Bits 31:24 MM[7:0]: Message Marker**

Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.

**E1 Bit 23:22 ET[1:0]: Event Type**

00: Reserved

01: Tx event

10: Transmission in spite of cancellation (always set for transmissions in DAR mode)

11: Reserved

**E1 Bit 21 EDL: Extended Data Length**

0: Standard frame format

1: CAN FD frame format (new DLC-coding and CRC)

**E1 Bit 20 BRS: Bit Rate Switch**

0: Frame transmitted without bit rate switching

1: Frame transmitted with bit rate switching

Bosch CAN-FD Specification V1.0

**E1 Bits 19:16 DLC[3:0]: Data Length Code**

0 to 8: CAN + CAN FD: frame with 0-8 data bytes transmitted

9 to 15: CAN: frame with 8 data bytes transmitted

9 to 15: CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted

**E1 Bits 15:0 TXTS[15:0]:Tx Timestamp**

Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler MTTCAN0TSCC.TCP.

Bosch CAN-FD Specification V1.0

### (5) Standard Message ID Filter Element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address MTTCAN0SIDFC.FLSSA plus the index of the filter element (0 to 127).

**Table 37.135 Standard Message ID Filter Element**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | SFT[1:0] | | SFEC[2:0] | | SFID1[10:0] | | | | | | | | | | | res | | | | | | | | SFID2[10:0] | | | | | | | |

**Bits 31:30 SFT[1:0]: Standard Filter Type**

00: Range filter from SF1ID to SF2ID (SF2ID ≥ SF1ID)

01: Dual ID filter for SF1ID or SF2ID

10: Classic filter: SF1ID = filter, SF2ID = mask

11: Reserved

**Bit 29:27 SFEC[2:0]:Standard Filter Element Configuration**

All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag MTTCAN0IR.HPM and, if enabled, an interrupt is generated. In this case register MTTCAN0HPMS is updated with the status of the priority match.

000: Disable filter element

001: Store in Rx FIFO 0 if filter matches

010: Store in Rx FIFO 1 if filter matches

011: Reject ID if filter matches

100: Set priority if filter matches

101: Set priority and store in FIFO 0 if filter matches

110: Set priority and store in FIFO 1 if filter matches

111: Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored

**Bits 26:16 SFID1[10:0]:Standard Filter ID 1**

First ID of standard ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.

Bosch CAN FD Specification V1.0

**Bits 10:0 SFID2[10:0]:Standard Filter ID 2**

This bit field has a different meaning depending on the configuration of SFEC:

1) SFEC = “001” to ”110” Second ID of standard ID filter element

2) SFEC = “111” Filter for Rx Buffers or for debug messages

**SFID2[10:9]**: These bits decide whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.
00: Store message into an Rx Buffer
01: Debug Message A
10: Debug Message B
11: Debug Message C

**SFID2[8:6]**: These bits are used to control the filter event pins m_ttcan_fe[2:0] at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_ttcan_hclk (CLK_HSB) period in case the filter matches.

**SFID2[5:0]:** These bits define the offset to the Rx Buffer Start Address MTTCAN0RXBC.RBSA for storage of a matching message.

Bosch CAN-FD Specification V1.0

### (6) Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address MTTCAN0XIDFC.FLESA plus two times the index of the filter element (0 to 63).

**Table 37.136 Extended Message ID Filter Element**

| | 31 | 24 | 23 | 16 | 15 | 8 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|
| F0 | EFEC[2:0] | | EFID1[28:0] | | | | | |
| F1 | EFT[1:0] | res | EFID2[28:0] | | | | | |

#### F0 Bit 31:29 EFEC[2:0]:Extended Filter Element Configuration

All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = “100”, “101”, or “110” a match sets interrupt flag MCANnIR.HPM and, if enabled, an interrupt is generated. In this case register MCANnHPMS is updated with the status of the priority match.

000: Disable filter element

001: Store in Rx FIFO 0 if filter matches

010: Store in Rx FIFO 1 if filter matches

011: Reject ID if filter matches

100: Set priority if filter matches

101: Set priority and store in FIFO 0 if filter matches

110: Set priority and store in FIFO 1 if filter matches

111: Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored

Bosch CAN-FD Specification V1.0

#### F0 Bits 28:0 EFID1[28:0]:Extended Filter ID 1

First ID of extended ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only MTTCAN0XIDAM masking mechanism (see **Section (e), Extended Message ID Filtering**) is used.

#### F1 Bits 31:30 EFT[1:0]: Extended Filter Type

00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID)

01: Dual ID filter for EF1ID or EF2ID

10: Classic filter: EF1ID = filter, EF2ID = mask

11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), MTTCAN0XIDAM mask not applied

#### F1 Bits 28:0 EFID2[28:0]:Extended Filter ID 2

This bit field has a different meaning depending on the configuration of EFEC:

1) EFEC: “001” to ”110”Second ID of extended ID filter element

2) EFEC: “111”Filter for Rx Buffers or for debug messages

**EFID2[10:9]**: These bits decide whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.
00: Store message into an Rx Buffer
01: Debug Message A
10: Debug Message B
11: Debug Message C

**EFID2[8:6]**: These bits are used to control the filter event pins m_ttcan_fe[2:0] at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_ttcan_hclk (CLK_HSB) period in case the filter matches.

**EFID2[5:0]:** These bits define the offset to the Rx Buffer Start Address MCANnRXBC.RBSA for storage of a matching message.

Bosch CAN-FD Specification V1.0

### (7) Trigger Memory Element

Up to 64 trigger memory elements can be configured. When accessing a Trigger Memory element, its address is the Trigger Memory Start Address MTTCAN0TTTMC.TMSA plus the index of the trigger memory element (0 to 63).

Table 37.137 Trigger Memory Element

| | 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|---|
| T0 | TM[15:0] | | res | CC[6:0] | ASC[1:0] | TMIN | TMEX | TYPE[3:0] |
| T1 | res | FTYPE | MNR[6:0] | res | MSC[2:0] |

#### T0 Bit 31:16 TM[15:0]:Time Mark

Cycle time for which the trigger becomes active.

#### T0 Bit 14:8 CC[6:0]:Cycle Code

Cycle count for which the trigger is valid. Ignored for trigger types Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, End_of_List.

$000000x_B$ valid for all cycles

$000001c_B$ valid every 2nd cycle at cycle count mod2 = c

$00001cc_B$ valid every 4th cycle at cycle count mod4 = cc

$0001ccc_B$ valid every 8th cycle at cycle count mod8 = ccc

$001cccc_B$ valid every 16th cycle at cycle count mod16 = cccc

$01ccccc_B$ valid every 32nd cycle at cycle count mod32 = ccccc

$1cccccc_B$ valid every 64th cycle at cycle count mod64 = cccccc

Bosch CAN-FD Specification V1.0

#### T0 Bit 7:6 ASC[1:0]:Asynchronous Serial Communication

00: No ASC operation

01: Reserved, do not use

10: Node is ASC receiver

11: Node is ASC transmitter

#### T0 Bit 5 TMIN:Time Mark Event Internal

0: No action

1: MTTCAN0TTIR.TTMI is set when trigger memory element becomes active

#### T0 Bit 4 TMEX:Time Mark Event External

0: No action

1: Pulse at output m_ttcan_tmp (MTTCAN0TMP) with the length of one m_ttcan_cclk (CLKP_H2) period is generated when the time mark of the trigger memory element becomes active and MTTCAN0TTOCN.TTMIE = '1'

**T0 Bit 3:0 TYPE[3:0]:Trigger Type**

| | |
|---|---|
| 0000 | Tx_Ref_Trigger - valid when not in Gap |
| 0001 | Tx_Ref_Trigger_Gap - valid when in Gap |
| 0010 | Tx_Trigger_Single - starts a single transmission in an exclusive time window |
| 0011 | Tx_Trigger_Continuous - starts continuous transmission in an exclusive time window |
| 0100 | Tx_Trigger_Arbitration - starts a transmission in an arbitrating time window |
| 0101 | Tx_Trigger_Merged - starts a merged arbitration window |
| 0110 | Watch_Trigger - valid when not in Gap |
| 0111 | Watch_Trigger_Gap - valid when in Gap |
| 1000 | Rx_Trigger - check for reception |
| 1001 | Time_Base_Trigger - only control TMIN, TMEX, and ASC |
| 1010 to 1111 | End_of_List - illegal type, causes config error |

**NOTE**

For ASC operation (ASC = “10”, “11”) only trigger types Rx_Trigger and Time_Base_Trigger should be used.

**T1 Bit 23 FTYPE:Filter Type**

*0*= 11-bit standard message ID
*1*= 29-bit extended message ID

**T1 Bit 22:16 FTYPE:Filter Type**

Transmission: Trigger is valid for configured Tx Buffer number. Valid values are 0 to 31.
Reception: Trigger is valid for standard / extended message ID filter element number. Valid values are 0 to 63 resp. 0 to 127.

**T1 Bits 2:0 MSC[2:0]:**

Message Status Count

Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1).

0-7: Actual status

**NOTES**

1. The trigger memory elements have to be written when the M_TTCAN is in INIT state. Write access to the trigger memory elements outside INIT state is not allowed.
2. There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx_Ref_Trigger. In this case they become active at the time mark modified by the actual Reference Trigger Offset (MTTCAN0TTOST.RTO).

Bosch CAN-FD Specification V1.0

## 37.6.3 Functional Description

### 37.6.3.1 Operating Modes

#### (1) Software Initialization

Software initialization is started by setting bit MTTCAN0CCCR.INIT, either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While MTTCAN0CCCR.INIT is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output m_ttcan_tx is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting MTTCAN0CCCR.INIT does not change any configuration register. Resetting MTTCAN0CCCR.INIT finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (≡ Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_TTCAN configuration registers is only enabled when both bits MTTCAN0CCCR.INIT and MTTCAN0CCCR.CCE are set (protected write).

MTTCAN0CCCR.CCE can only be set/reset while MTTCAN0CCCR.INIT = '1'.
MTTCAN0CCCR.CCE is automatically reset when MTTCAN0CCCR.INIT is reset.

The following registers are reset when MTTCAN0CCCR.CCE is set

- MTTCAN0HPMS - High Priority Message Status
- MTTCAN0RXF0S - Rx FIFO 0 Status
- MTTCAN0RXF1S - Rx FIFO 1 Status
- MTTCAN0TXFQS - Tx FIFO/Queue Status
- MTTCAN0TXBRP - Tx Buffer Request Pending
- MTTCAN0TXBTO - Tx Buffer Transmission Occurred
- MTTCAN0TXBCF - Tx Buffer Cancellation Finished
- MTTCAN0TXEFS - Tx Event FIFO Status
- MTTCAN0TTOST - TT Operation Status
- MTTCAN0TTLGT - TT Local & Global Time, only Global Time MTTCAN0TTLGT.GT is reset
- MTTCAN0TTCTC - TT Cycle Time & Count
- MTTCAN0TTCSM - TT Cycle Sync Mark

The Timeout Counter value MTTCAN0TOCV.TOC is preset to the value configured by MTTCAN0TOCC.TOP when MTTCAN0CCCR.CCE is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while MTTCAN0CCCR.CCE = '1'.

The following registers are only writable while MTTCAN0CCCR.CCE = '0'

- MTTCAN0TXBAR - Tx Buffer Add Request
- MTTCAN0TXBCR - Tx Buffer Cancellation Request

MTTCAN0CCCR.TEST and MTTCAN0CCCR.MON can only be set by the Host while MTTCAN0CCCR.INIT = '1' and MTTCAN0CCCR.CCE = '1'. Both bits may be reset at any time.
MTTCAN0CCCR.DAR can only be set/reset while MTTCAN0CCCR.INIT = '1' and MTTCAN0CCCR.CCE = '1'.

Bosch CAN-FD Specification V1.0

### (2) Normal Operation

The M_TTCAN's default operating mode after hardware reset is event-driven CAN communication without time triggers (MTTCAN0TTOCF.OM = "00"). It is required that both MTTCAN0CCCR.INIT and MTTCAN0CCCR.CCE are set before the TT Operation Mode can be changed.

Once the M_TTCAN is initialized and MTTCAN0CCCR.INIT is reset to zero, the M_TTCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

### (3) CAN FD Operation

There are two variants in the CAN FD frame format, first the CAN FD frame without bit rate switching where the data field of a CAN frame may be longer than 8 bytes. The second variant is the CAN FD frame where control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame.

The CAN operation mode is enabled by programming MTTCAN0CCCR.CME. In case MTTCAN0CCCR.CME = "01" transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With MTTCAN0CCCR.CME = "10"/"11" transmission and reception of long and fast CAN FD frames is enabled. MTTCAN0CCCR.CME can only be changed while MTTCAN0CCCR.INIT and MTTCAN0CCCR.CCE are both set.

When initialization is left (MTTCAN0CCCR.INIT set to '0'), the CAN FD protocol option is inactive, it has to be requested by writing to MTTCAN0CCCR.CMR.

A mode change requested by writing to MTTCAN0CCCR.CMR will be executed next time the CAN protocol controller FSM reaches idle phase between CAN frames. Upon this event MTTCAN0CCCR.CMR is reset to "00" and the status flags MTTCAN0CCCR.FDBS and MTTCAN0CCCR.FDO are set accordingly. In case the requested CAN operation mode is not enabled, the value written to MTTCAN0CCCR.CMR is retained until it is overwritten by the next mode change request. Default is CAN operation according to ISO11898-1.

It is not necessary to change the CAN operation mode after system startup. A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting according to ISO11898-1 until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in silent mode until programming has completed. Then all nodes switch back to CAN communication according ISO11898-1.

When MTTCAN0CCCR.CME ≠ "00", received CAN FD frames are interpreted according to the CAN FD Protocol Specification. The reserved bit in CAN frames with 11-bit identifiers and the first reserved bit in CAN frames with 29-bit identifiers will be decoded as EDL bit. EDL = recessive signifies a CAN FD frame, EDL = dominant signifies a standard CAN frame. In a CAN FD frame, the two bits following EDL, r0 and BRS, decide whether the bit rate inside of this CAN FD frame is

Bosch CAN FD Specification V1.0

switched. A CAN FD bit rate switch is signified by r0 = dominant and BRS = recessive. The coding of r0 = recessive is reserved for future expansion of the protocol.

Reception of CAN frames according to ISO 11898-1 is possible in all CAN operation modes.

The status bits MTTCAN0CCCR.FDO and MTTCAN0CCCR.FDBS indicate the format of transmitted frames. When MTTCAN0CCCR.FDO is set, frames will be transmitted in CAN FD format with EDL = recessive. When both MTTCAN0CCCR.FDO and MTTCAN0CCCR.FDBS are set, frames will be transmitted in CAN FD format with bit rate switching and both bits EDL and BRS = recessive.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to **Table 37.138** below.

**Table 37.138 Coding of DLC in CAN FD**

| DLC | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| Number of Data Bytes | 12 | 16 | 20 | 24 | 32 | 48 | 64 |

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing & Prescaler Register MTTCAN0BTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the Fast Bit Timing & Prescaler Register MTTCAN0FBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (m_ttcan_cclk: CLKP_H2). Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

Bosch CAN FD Specification V1.0

### (4) Transmitter Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin m_ttcan_tx the protocol controller receives the transmitted data from its local CAN transceiver via pin m_ttcan_rx. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transceiver loop delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

(a) Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver. **Figure 37.16** below describes how the transceiver loop delay is measured.

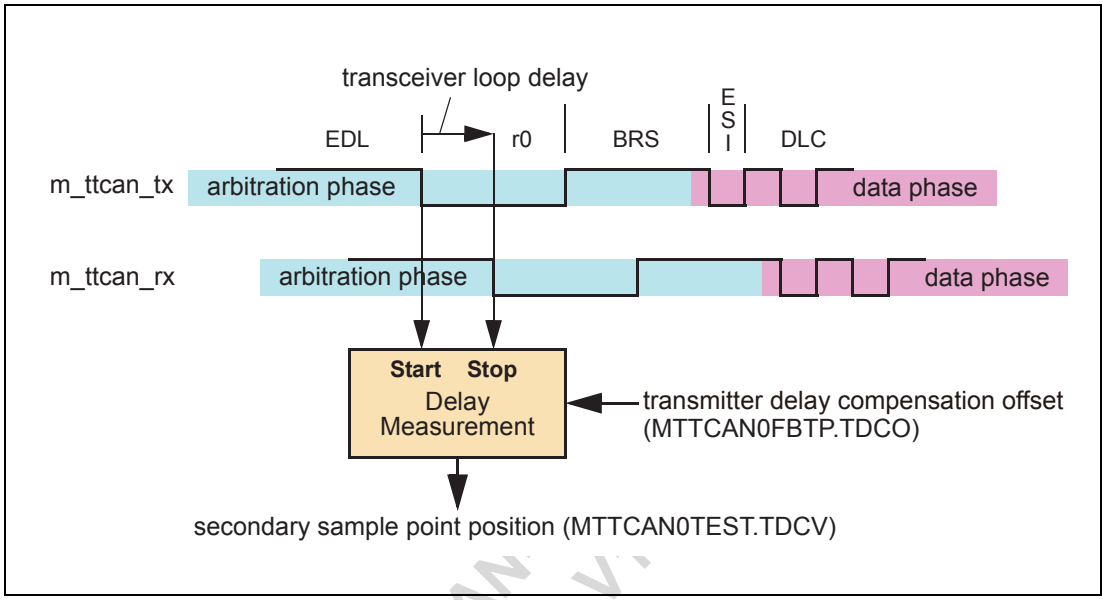



**Figure 37.16 Transmitter delay measurement**

Within each CAN FD frame, the transmitter measures the delay between the data transmitted at pin m_ttcan_tx and the data received at pin m_ttcan_rx. The measurement is done once, at the falling edge of bit EDL to bit r0. The delay is measured in m_ttcan_cclk (CLKP_H2) periods.

A secondary sample point position is calculated by adding a configurable transmitter delay compensation offset MTTCAN0FBTP.TDCO to the measured transceiver delay. This transmitter delay compensation value MTTCAN0TEST.TDCV is the sum of the measured transceiver delay and the transmitter delay compensation offset. The transmitter delay compensation offset is chosen to adjust the secondary sample point inside the bit time (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of time quanta $t_q$.

To check for bit errors during the data phase, the delayed transmit data is compared against the received data at the secondary sample point. If a bit error is detected at the secondary sample point, the transmitter will react to this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

For the transmitter delay compensation the following boundary conditions have to be considered:

- The sum of the measured delay from m_ttcan_tx to m_ttcan_rx and the configured transmitter delay compensation offset MTTCAN0FBTP.TDCO has to be less than 3 bit times in the data phase.
- The sum of the measured delay from m_ttcan_tx to m_ttcan_rx and the configured transmitter delay compensation offset MTTCAN0FBTP.TDCO has to be less or equal 63 m_ttcan_cclk (CLKP_H2) periods. In case this sum exceeds 63 m_ttcan_cclk (CLKP_H2) periods, the maximum value of 63 m_ttcan_cclk (CLKP_H2) periods is used for transmitter delay compensation.

The actual delay compensation value is monitored by reading MTTCAN0TEST.TDCV.

(b) Configuration and Status

Compensation for the transceiver loop delay by the M_TTCAN is enabled via MTTCAN0FBTP.TDC. The transmitter delay compensation offset is configured via MTTCAN0FBTP.TDCO. The actual delay compensation value applied by the M_TTCAN's protocol engine can be read from MTTCAN0TEST.TDCV.

Bosch CAN-FD Specification V1.0

### (5) Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. The Host can set the M_TTCAN into Restricted Operation mode by setting bit MTTCAN0CCCR.ASM. The bit can only be set by the Host when both MTTCAN0CCCR.CCE and MTTCAN0CCCR.INIT are set to '1'. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset MTTCAN0CCCR.ASM.

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

When the M_TTCAN is configured for Asynchronous Serial Communication, the Host has to set MTTCAN0CCCR.ASM during initialization to start with an ASC window. The bit is reset at the end of the first ASC window after the M_TTCAN has finished initializing. During time-triggered operation MTTCAN0CCCR.ASM is set by the M_TTCAN at the beginning of an ASC window (trigger memory element with T0.ASC = "10", "11"). It is reset by each trigger memory element with T0.ASC = "00".

If the M_TTCAN is connected to a Clock Calibration on CAN unit, MTTCAN0CCCR.ASM is controlled by input m_ttcan_cok. In case m_ttcan_cok switches to '0', bit MTTCAN0CCCR.ASM is set. When m_ttcan_cok switches back to '1', bit MTTCAN0CCCR.ASM returns to the previously written value. The state of MTTCAN0CCCR.ASM is the written value while input m_ttcan_cok is at '1'. The input is hardwired to '1' when there is no Clock Calibration on CAN unit connected.

Bosch CAN FD Specification V1.0

### (6) Bus Monitoring Mode

The M_TTCAN is set in Bus Monitoring Mode by programming MTTCAN0CCCR.MON to one or when error level S3 (MTTCAN0TTOST.EL = “11”) is entered. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the M_TTCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the M_TTCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_TTCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode register MTTCAN0TXBRP is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. **Figure 37.17** shows the connection of signals m_ttcan_tx and m_ttcan_rx to the M_TTCAN in Bus Monitoring Mode.

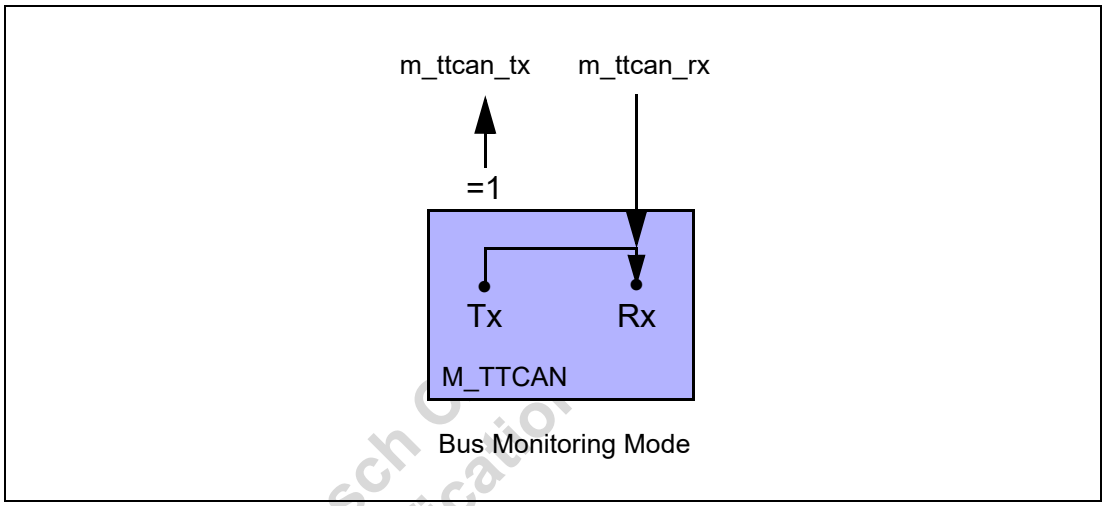



Figure 37.17 Pin Control in Bus Monitoring Mode

Bosch CAN Specification

### (7) Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the M_TTCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via MTTCAN0CCCR.DAR.

#### (a) Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer’s Tx Request Pending bit MTTCAN0TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  Corresponding Tx Buffer Transmission Occurred bit MTTCAN0TXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit MTTCAN0TXBCF.CFx not set
- Successful transmission in spite of cancellation:
  Corresponding Tx Buffer Transmission Occurred bit MTTCAN0TXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit MTTCAN0TXBCF.CFx set
- Arbitration lost or frame transmission disturbed:
  Corresponding Tx Buffer Transmission Occurred bit MTTCAN0TXBTO.TOx not set

Corresponding Tx Buffer Cancellation Finished bit MTTCAN0TXBCF.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = “10” (transmission in spite of cancellation).

### (8) Power Down (Sleep Mode)

The M_TTCAN can be set into power down mode by using CC Control Register MTTCAN0CCCR.CSR. When all pending transmission requests have completed, the M_TTCAN waits until bus idle state is detected. Then the M_TTCAN sets then MTTCAN0CCCR.INIT to one to prevent any further CAN transfers. Now the M_TTCAN acknowledges that it is ready for power down by MTTCAN0CCCR.CSA to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to MTTCAN0CCCR.INIT will have no effect. Now the module clock inputs m_ttcan_hclk (CLK_HSB) and m_ttcan_cclk (CLKP_H2) may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting CC Control Register flag MTTCAN0CCCR.CSR. The M_TTCAN will acknowledge this by resetting MTTCAN0CCCR.CSA. Afterwards, the application can restart CAN communication by resetting bit MTTCAN0CCCR.INIT.

### (9) Test Modes

To enable write access to register MTTCAN0TEST (see **(4) MCANnTEST — Test Register**), bit MTTCAN0CCCR.TEST has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin m_ttcan_tx by programming MTTCAN0TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_TTCAN’s bit timing and it can drive constant dominant or recessive values. The actual value at pin m_ttcan_rx can be read from MTTCAN0TEST.RX. Both functions can be used to check the CAN bus’ physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to MTTCAN0TEST.TX until the new configuration is visible at output pin m_ttcan_tx. This applies also when reading input pin m_ttcan_rx via MTTCAN0TEST.RX.

Bosch CAN-FD Specification V1.0

**NOTE**

Test modes should be used for production tests or self test only. The software control for pin m_ttcan_tx interferes with all CAN protocol functions. It is not recommended to use test modes for application.

#### (a) External Loop Back Mode

The M_TTCAN can be set in External Loop Back Mode by programming MTTCAN0TEST.LBCK to one. In Loop Back Mode, the M_TTCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx Buffer or an Rx FIFO. **Figure 37.18** shows the connection of signals m_ttcan_tx and m_ttcan_rx to the M_TTCAN in External Loop Back Mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_TTCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the M_TTCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the m_ttcan_rx input pin is disregarded by the M_TTCAN. The transmitted messages can be monitored at the m_ttcan_tx pin.

(b) Internal Loop Back Mode

Internal Loop Back Mode is entered by programming bits MTTCAN0TEST.LBCK and MTTCAN0CCCR.MON to one. This mode can be used for a “Hot Selftest”, meaning the M_TTCAN can be tested without affecting a running CAN system connected to the pins m_ttcan_tx and m_ttcan_rx. In this mode pin m_ttcan_rx is disconnected from the M_TTCAN and pin m_ttcan_tx is held recessive. **Figure 37.18** shows the connection of m_ttcan_tx and m_ttcan_rx to the M_TTCAN in case of Internal Loop Back Mode.

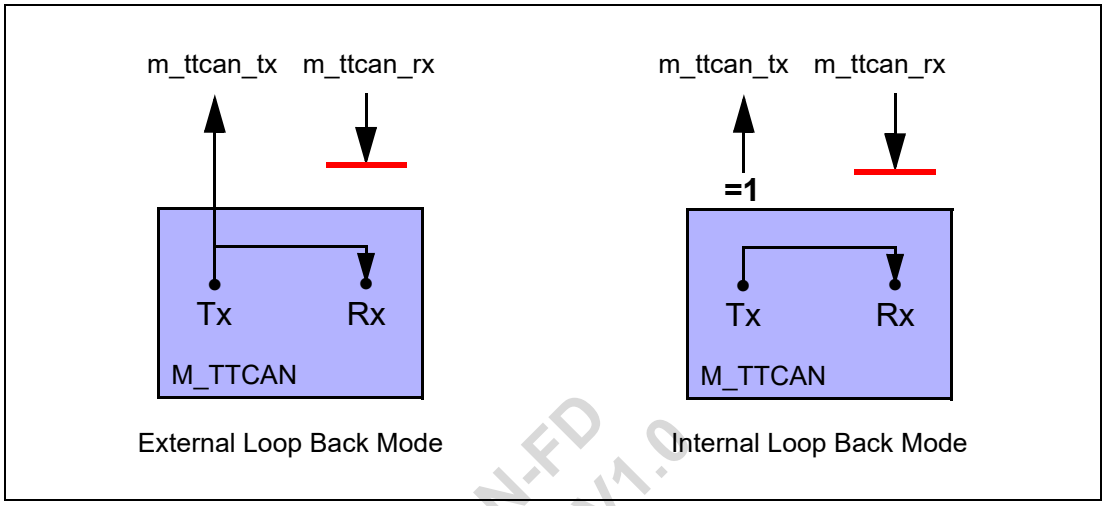



Figure 37.18 Pin Control in Loop Back Modes

Bosch CAN-FD Specification V1.0

### (10) Application Watchdog

The application watchdog is served by reading register MTTCAN0TTOST. When the application watchdog is not served in time, bit MTTCAN0TTOST.AWE is set, all TTCAN communication is stopped, and the M_TTCAN is set into Bus Monitoring Mode.

The TT Application Watchdog can be disabled by programming the Application Watchdog Limit MTTCAN0TTOCF.AWL to $00_H$. The TT Application Watchdog should not be disabled in a TTCAN application program.

### 37.6.3.2 Timestamp Generation

For timestamp generation the M_TTCAN supplies a 16-bit wrap-around counter. A prescaler MTTCAN0TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via MTTCAN0TSCV.TCV. A write access to register MTTCAN0TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag MTTCAN0IR.TSW is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer / Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit MTTCAN0TSCC.TSS an external 16-bit timestamp can be used.

### 37.6.3.3 Timeout Counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_TTCAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by MTTCAN0TSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via register MTTCAN0TOCC. The actual counter value can be read from MTTCAN0TOCV.TOC.

The Timeout Counter can only be started while MTTCAN0CCCR.INIT = '0'. It is stopped when MTTCAN0CCCR.INIT = '1', e.g. when the M_TTCAN enters Bus_Off state.

The operation mode is selected by MTTCAN0TOCC.TOS. When operating in Continuous Mode, the counter starts when MTTCAN0CCCR.INIT is reset. A write to MTTCAN0TOCV presets the counter to the value configured by MTTCAN0TOCC.TOP and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by MTTCAN0TOCC.TOP. Down-counting is started when the first FIFO element is stored. Writing to MTTCAN0TOCV has no effect.

When the counter reaches zero, interrupt flag MTTCAN0IR.TOO is set. In Continuous Mode, the counter is immediately restarted at MTTCAN0TOCC.TOP.

Bosch CAN-FD Specification V1.0

**NOTE**

The clock signal for the Timeout Counter is derived from the CAN Core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

### 37.6.3.4 Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

#### (1) Acceptance Filtering

The M_TTCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration MTTCAN0GFC
- Standard ID Filter Configuration MTTCAN0SIDFC
- Extended ID Filter Configuration MTTCAN0XIDFC
- Extended ID AND Mask MTTCAN0XIDAM

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag MTTCAN0IR.HPM
- Set High Priority Message interrupt flag MTTCAN0IR.HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

Bosch CAN FD Specification V1.0

**Rx Buffer**

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see MTTCAN0PSR.LEC respectively MTTCAN0PSR.FLEC.

**Rx FIFO**

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see MTTCAN0PSR.LEC respectively MTTCAN0PSR.FLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in **(b) Rx FIFO Overwrite Mode** have to be considered.

**NOTE**

When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

(a) Range Filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

EFT: “00”: The Message ID of received frames is ANDed with the Extended ID AND Mask (MTTCAN0XIDAM) before the range filter is applied

EFT: “11”: The Extended ID AND Mask (MTTCAN0XIDAM) is not used for range filtering

(b) Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

(c) Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

Bosch CAN-FD Specification V1.0

(d) Standard Message ID Filtering

**Figure 37.19** below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in **(5) Standard Message ID Filter Element**.

Controlled by the Global Filter Configuration MTTCAN0GFC and the Standard ID Filter Configuration MTTCAN0SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

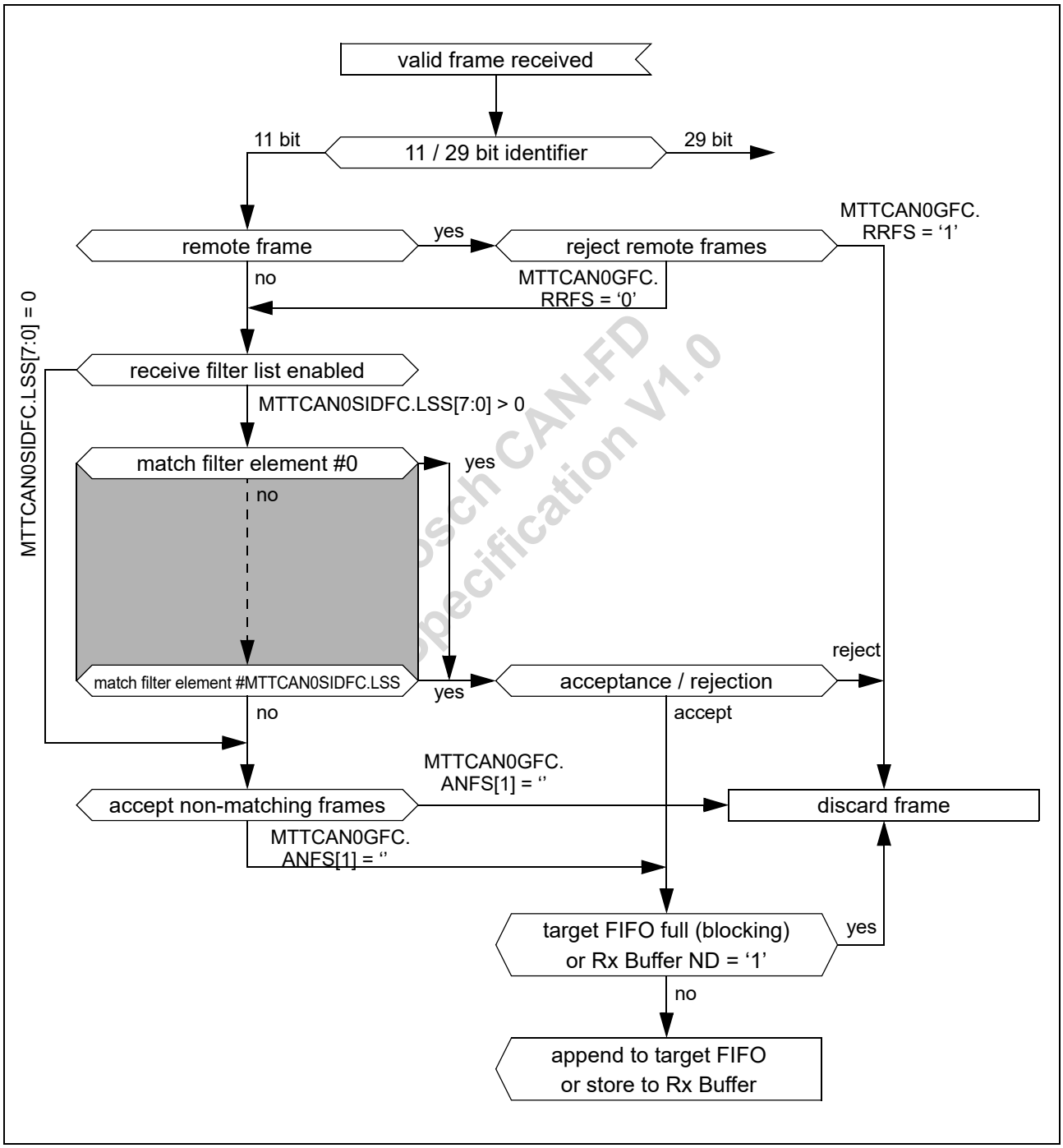



**Figure 37.19 Standard Message ID Filter Path**

(e) Extended Message ID Filtering

**Figure 37.20** below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in **(6) Extended Message ID Filter Element**.

Controlled by the Global Filter Configuration MTTCAN0GFC and the Extended ID Filter Configuration MTTCAN0XIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask MTTCAN0XIDAM is ANDed with the received identifier before the filter list is executed.

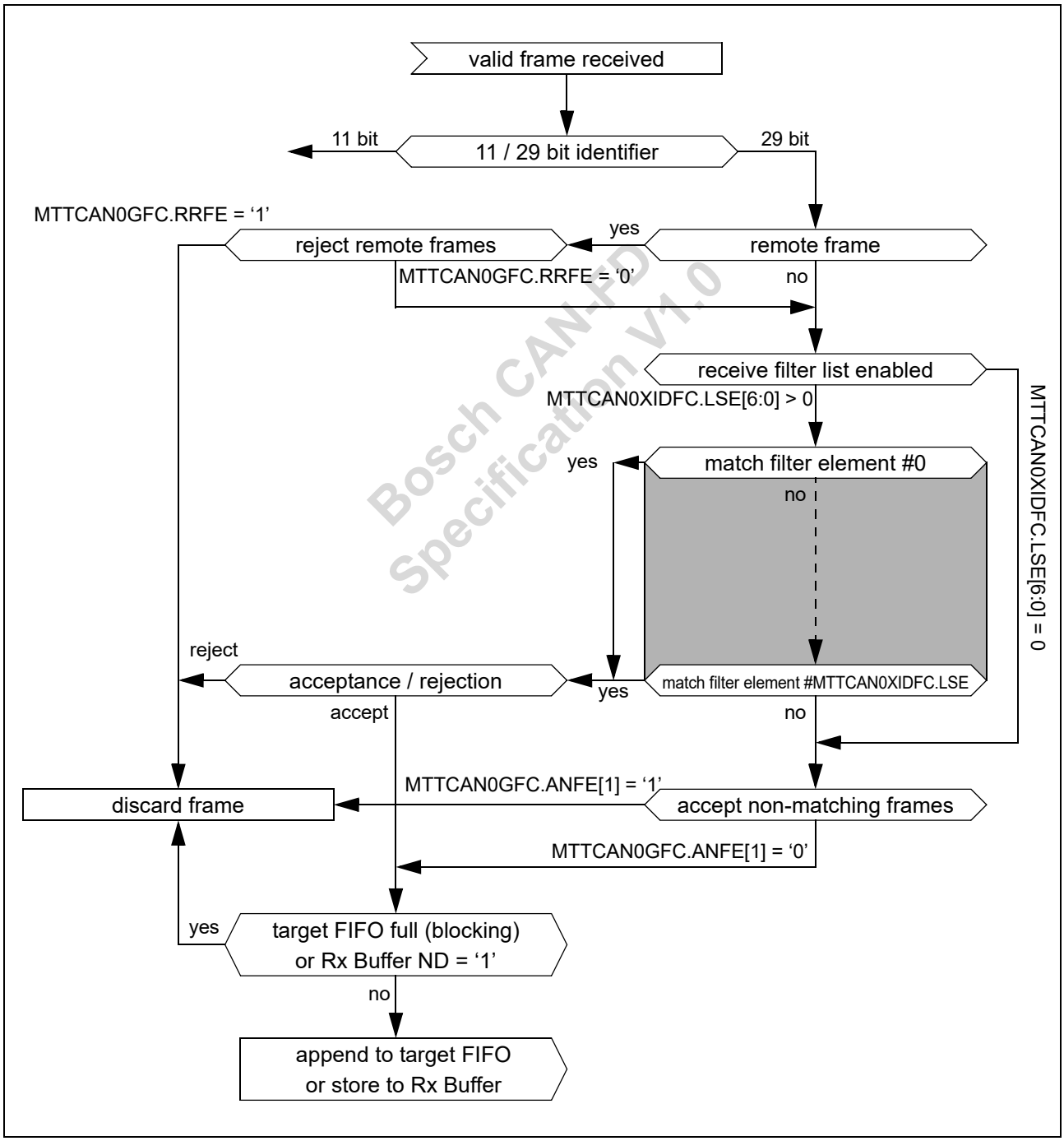



**Figure 37.20 Extended Message ID Filter Path**

### (2) Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers MTTCAN0RXF0C and MTTCAN0RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see **(1) Acceptance Filtering**. The Rx FIFO element is described in **(2) Rx Buffer and FIFO Element**.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag MTTCAN0IR.RFnW is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS.FnF. In addition interrupt flag MTTCAN0IR.RFnF is set.

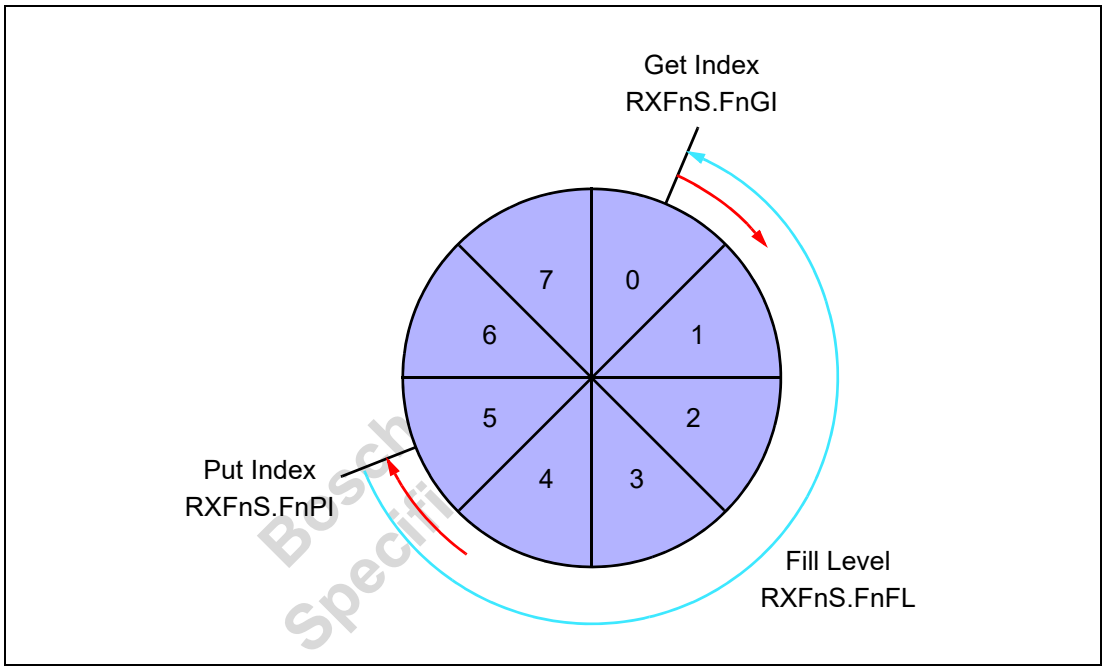



**Figure 37.21 Rx FIFO Status**

When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI • FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

**Table 37.139 Rx Buffer / FIFO Element Size**

| MTTCAN0RXESC.RBDS[2:0]<br>MTTCAN0RXESC.FnDS[2:0] | Data Field<br>[bytes] | FIFO Element Size<br>[RAM words] |
|---|---|---|
| 000 | 8 | 4 |
| 001 | 12 | 5 |
| 010 | 16 | 6 |
| 011 | 20 | 7 |
| 100 | 24 | 8 |
| 101 | 32 | 10 |
| 110 | 48 | 14 |
| 111 | 64 | 18 |

(a) Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = '0'. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS.FnF = '1'. In addition interrupt flag MTTCAN0IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = '1'. In addition interrupt flag MTTCAN0IR.RFnL is set.

(b) Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = '1'.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signalled by RXFnS.FnF = '1', the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. **Figure 37.22** shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

Bosch CAN FD Specification V1.0

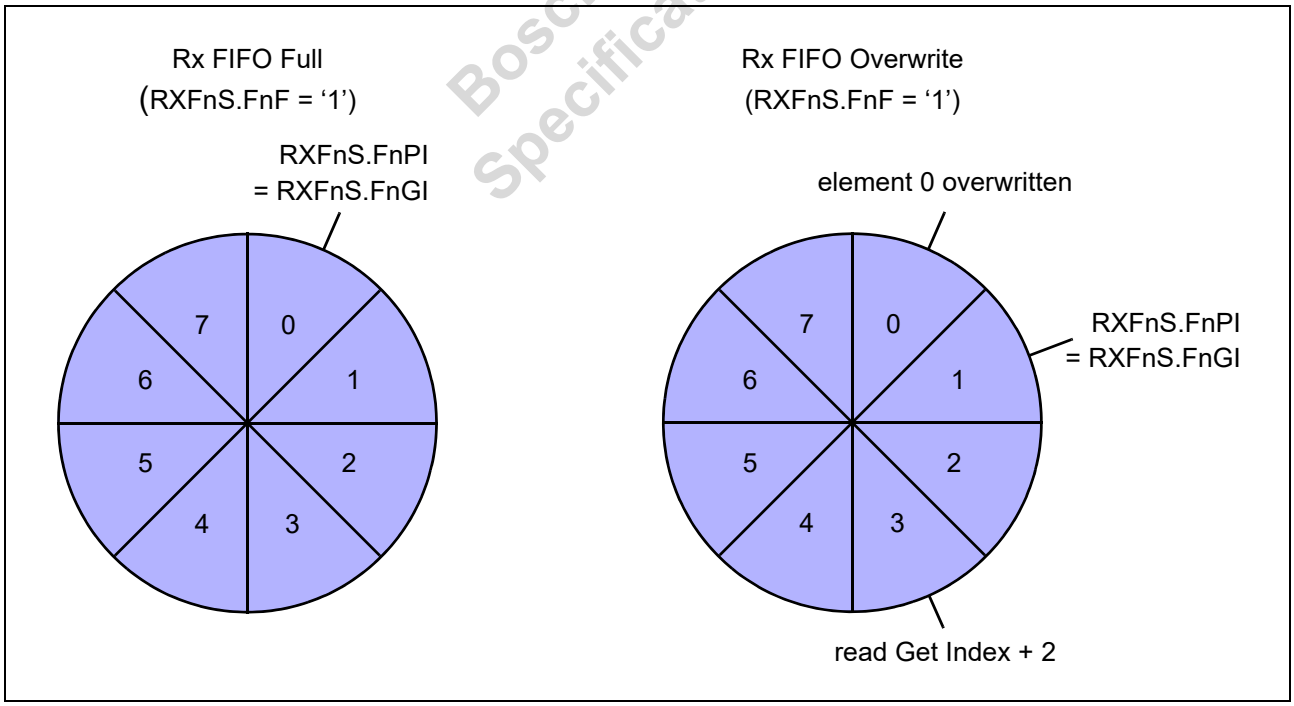



**Figure 37.22 Rx FIFO Overflow Handling**

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = '0').

### (3) Dedicated Rx Buffers

The M_TTCAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via MTTCAN0RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = “111” and SFID2 / EFID2[10:9] = “00” has to be configured (see **(5) Standard Message ID Filter Element** and **(6) Extended Message ID Filter Element**).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag MTTCAN0IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

**Table 37.140 Example Filter Configuration for Rx Buffers**

| Filter Element | SFID1[10:0] EFID1[28:0] | SFID2[10:9] EFID2[10:9] | SFID2[5:0] EFID2[5:0] |
|---|---|---|---|
| 0 | ID message 1 | 00 | 00 0000 |
| 1 | ID message 2 | 00 | 00 0001 |
| 2 | ID message 3 | 00 | 00 0010 |

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register MTTCAN0NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a ‘1’ to the respective bit position.

While an Rx Buffer’s New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

Bosch CAN FD Specification V1.0

#### (a) Rx Buffer Handling

- Reset interrupt flag MTTCAN0IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

### 37.6.3.5 Tx Handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in **(3) Tx Buffer Element**.

**NOTE**

AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register MTTCAN0TXBRP is updated, or when a transmission has been started.

#### (1) Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

Bosch CAN-FD Specification V1.0

The transmit pause feature is controlled by bit MTTCAN0CCCR.TXP. If the bit is set, the M_TTCAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (MTTCAN0CCCR.TXP = '0').

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

#### (2) Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via MTTCAN0TXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (see **Table 37.141**). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0 to 31) • Element Size to the Tx Buffer Start Address MTTCAN0TXBC.TBSA.

Table 37.141 Tx Buffer / FIFO / Queue Element Size

| MTTCAN0TXESC.TBDS[2:0] | Data Field [bytes] | Element Size [RAM words] |
|---|---|---|
| 000 | 8 | 4 |
| 001 | 12 | 5 |
| 010 | 16 | 6 |
| 011 | 20 | 7 |
| 100 | 24 | 8 |
| 101 | 32 | 10 |
| 110 | 48 | 14 |
| 111 | 64 | 18 |

### (3) Tx FIFO

Tx FIFO operation is configured by programming MTTCAN0TXBC.TFQM to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index MTTCAN0TXFQS.TFGI. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_TTCAN calculates the Tx FIFO Free Level MTTCAN0TXFQS.TFFL as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index MTTCAN0TXFQS.TFQPI. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (MTTCAN0TXFQS.TFQF = '1') is signalled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the MTTCAN0TXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via MTTCAN0TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see **Table 37.141**). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index MTTCAN0TXFQS.TFQPI (0 to 31) • Element Size to the Tx Buffer Start Address MTTCAN0TXBC.TBSA.

Bosch CAN FD Specification V1.0

### (4) Tx Queue

Tx Queue operation is configured by programming MTTCAN0TXBC.TFQM to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index MTTCAN0TXFQS.TFQPI. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (MTTCAN0TXFQS.TFQF = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register MTTCAN0TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see **Table 37.141**). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index MTTCAN0TXFQS.TFQPI (0 to 31) • Element Size to the Tx Buffer Start Address MTTCAN0TXBC.TBSA.

### (5) Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by MTTCAN0TXBC.NDTB. The number of Tx Buffers assigned to the Tx FIFO is configured by MTTCAN0TXBC.TFQS. In case MTTCAN0TXBC.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

Bosch CAN FD Specification V1.0

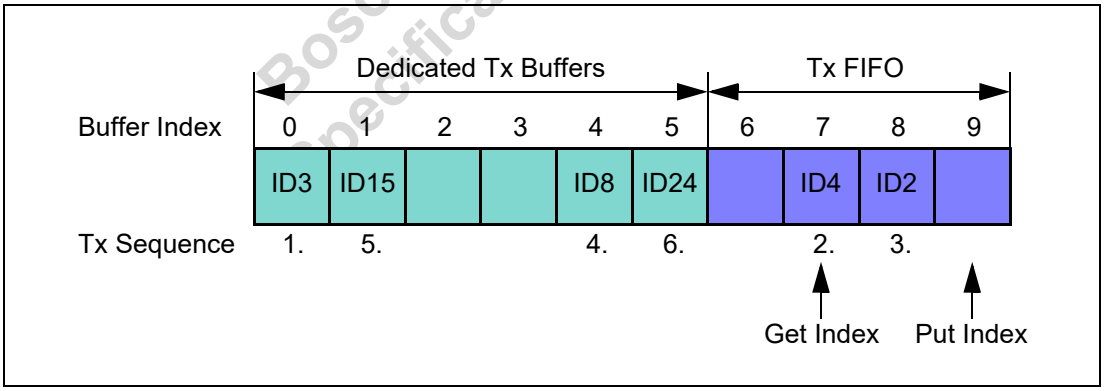



**Figure 37.23 Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO**

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS.TFGI)
- Buffer with lowest Message ID gets highest priority and is transmitted next

### (6) Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by MTTCAN0TXBC.NDTB. The number of Tx Queue Buffers is configured by MTTCAN0TXBC.TFQS. In case MTTCAN0TXBC.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

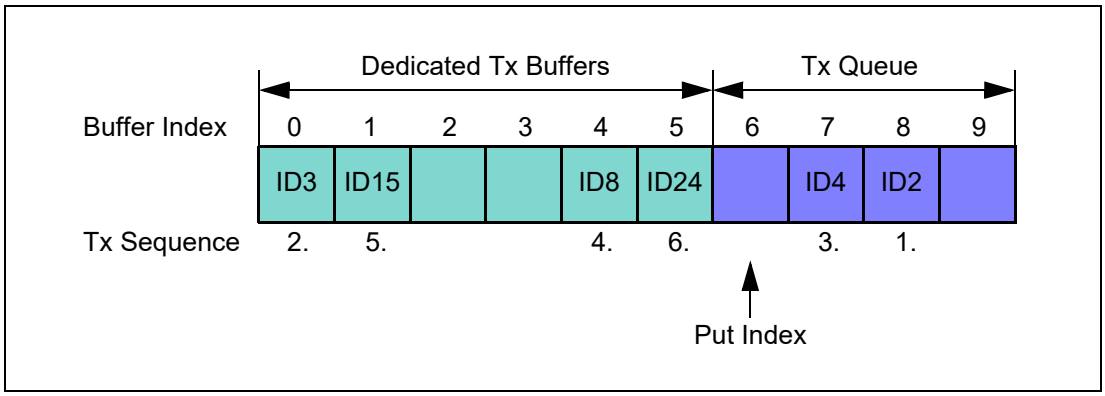



**Figure 37.24 Example of mixed Configuration Dedicated Tx Buffers / Tx Queue**

Bosch CAN-FD Specification V1.0

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

### (7) Transmit Cancellation

The M_TTCAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a Dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register MTTCAN0TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signalled by setting the corresponding bit of register MTTCAN0TXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding MTTCAN0TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding MTTCAN0TXBTO and MTTCAN0TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding MTTCAN0TXBCF bit is set.

**NOTE**

In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

### (8) Tx Event Handling

To support Tx event handling the M_TTCAN has implemented a Tx Event FIFO. After the M_TTCAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in **(4) Tx Event FIFO Element**.

When a Tx Event FIFO full condition is signalled by MTTCAN0IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag MTTCAN0IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by MTTCAN0TXEFC.EFWM, interrupt flag MTTCAN0IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index MTTCAN0TXEFS.EFGI has to be added to the Tx Event FIFO start address MTTCAN0TXEFC.EFSA.

Bosch CAN-FD Specification V1.0

#### 37.6.3.6 FIFO Acknowledge Handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see **(27) MTTCAN0RXF0A — Rx FIFO 0 Acknowledge**, **(31) MTTCAN0RXF1A — Rx FIFO 1 Acknowledge**, and **(45) MTTCAN0TXEFA — Tx Event FIFO Acknowledge**). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.

When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the M_TTCAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

**NOTE**

The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_TTCAN does not check for erroneous values.

Bosch CAN FD Specification V1.0

## 37.6.4 TTCAN Operation

### 37.6.4.1 Reference Message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the Time Master (sender of the reference message).

For Level 1 the data length must be at least one; for Level 0,2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to 8 potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register MTTCAN0TTRMC.

The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted Master_Ref_Mark is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (Next_is_Gap bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters.

The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

Bosch CAN-FD Specification V1.0

#### (1) Level 1

Level 1 operation is configured via MTTCAN0TTOCF.OM = “01” and MTTCAN0TTOCF.GEN. External clock synchronization is not available in Level 1.

The information related to the reference message is stored in the first data byte as shown in **Table 37.142** below. Cycle_Count is optional.

**Table 37.142 First byte of Level 1 reference message**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| First Byte | Next_is_Gap | res | Cycle_Count[5:0] | | | | | |

### (2) Level 2

Level 2 operation is configured via MTTCAN0TTOCF.OM = “10” and MTTCAN0TTOCF.GEN.

The information related to the reference message is stored in the first four data bytes as shown in **Table 37.143** below. Cycle_Count and the lower four bits of NTU_Res are optional. The M_TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

**Table 37.143 First four bytes of Level 2 reference message**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| First Byte | Next_is_Gap | res | Cycle_Count[5:0] | | | | | |
| Second Byte | NTU_Res[6:4] | | | | NTU_Res[3:0] | | | Disc_Bit |
| Third Byte | Master_Ref_Mark[7:0] | | | | | | | |
| Fourth Byte | Master_Ref_Mark[15:8] | | | | | | | |

### (3) Level 0

Level 0 operation is configured via MTTCAN0TTOCF.OM = “11”. External event-synchronized time-triggered operation is not available in Level 0.

The information related to the reference message is stored in the first four data bytes as shown in **Table 37.144** below. In Level 0 Next_is_Gap is always zero. Cycle_Count and the lower four bits of NTU_Res are optional. The M_TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

Bosch CAN-FD ecification V1.0

**Table 37.144 First four bytes of Level 0 reference message**

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| First Byte | Next_is_Gap | res | Cycle_Count[5:0] | | | | | |
| Second Byte | NTU_Res[6:4] | | | | NTU_Res[3:0] | | | Disc_Bit |
| Third Byte | Master_Ref_Mark[7:0] | | | | | | | |
| Fourth Byte | Master_Ref_Mark[15:8] | | | | | | | |

### 37.6.4.2 TTCAN Configuration

#### (1) TTCAN Timing

The Network Time Unit NTU is the unit in which all times are measured. The NTU is a constant of the whole network and is defined a priori by the network system designer. In TTCAN Level 1 the NTU is the nominal CAN bit time. In TTCAN Level 0 and Level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable.

In TTCAN Level 0 and Level 2 the length of the NTU is defined by the Time Unit Ratio TUR. The TUR is in principle a non-integer number and given by the formula TUR = MTTCAN0TURNA.NAV/ MTTCAN0TURCF.DC. The length of the NTU is given by the formula NTU = CAN Clock Period • TUR.

The TUR Numerator Configuration NC is an 18-bit number, MTTCAN0TURCF.NCL[15:0] can be programmed in the range $0000_H$ to $FFFF_H$. MTTCAN0TURCF.NCH[17:16] is hard wired to 0b01. When the number $nnnn_H$ is written to MTTCAN0TURCF.NCL[15:0], MTTCAN0TURNA.NAV starts with the value $10000_H + 0nnnn_H = 1nnnn_H$. The TUR Denominator Configuration MTTCAN0TURCF.DC is a 14-bit number. MTTCAN0TURCF.DC may be programmed in the range $0001_H$ to $3FFF_H$, $0000_H$ is an illegal value.

In Level 1, NC must be ≥ 4 • MTTCAN0TURCF.DC. In Level 0,2 NC must be ≥ 8 • MTTCAN0TURCF.DC to allow the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets MTTCAN0TURCF.DC to $1000_H$ and MTTCAN0TURCF.NCL to $10000_H$, resulting in an NTU consisting of 16 CAN clock periods. Local time and application watchdog are not started before either the MTTCAN0CCCR.INIT is reset, or MTTCAN0TURCF.ELT is set. MTTCAN0TURCF.ELT may not be set before the NTU is configured. Setting MTTCAN0TURCF.ELT to '1' also locks the write access to register MTTCAN0TURCF.

At startup MTTCAN0TURNA.NAV is updated from NC (= MTTCAN0TURCF.NCL + $10000_H$) when MTTCAN0TURCF.ELT is set. In TTCAN Level 1 there is no drift compensation. MTTCAN0TURNA.NAV does not change during operation, it always equals NC.

In TTCAN Level 0 and Level 2 there are two possibilities for MTTCAN0TURNA.NAV to change. When operating as time slave or backup time master, and when MTTCAN0TTOCF.ECC is set, MTTCAN0TURNA.NAV is updated automatically to the value calculated from the monitored global time speed, as long as the M_TTCAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization it returns to NC. When operating as the actual time master, and when MTTCAN0TTOCF.EECS is set, the Host may update MTTCAN0TURCF.NCL. When the Host sets MTTCAN0TTOCN.ECS, MTTCAN0TURNA.NAV will be updated from the new value of NC at the next reference message. The status flag MTTCAN0TTOST.WECS as is set when MTTCAN0TTOCN.ECS is set and is cleared when MTTCAN0TURNA.NAV is updated. MTTCAN0TURCF.NCL is write locked while MTTCAN0TTOST.WECS is set.

In TTCAN Level 0 and Level 2 the clock calibration process adapts MTTCAN0TURNA.NAV in the range of the Synchronization Deviation Limit SDL of NC ± $2^{(MTTCAN0TTOCF.LDSDL+5)}$. MTTCAN0TURCF.NCL should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of MTTCAN0TURNA.NAV.

The synchronization deviation SD is the difference between NC and MTTCAN0TURNA.NAV (SD = |NC - MTTCAN0TURNA.NAV|). It is limited by the Synchronization Deviation Limit SDL,

Bosch CAN-FD Specification V1.0

which is configured by its dual logarithm MTTCAN0TTOCF.LDSDL (SDL = $2^{(MTTCAN0TTOCF.LDSDL+5)}$) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new Basic Cycle. When the calculated MTTCAN0TURNA.NAV deviates by more than SDL from NC, or if the Disc_Bit in the reference message is set, the drift compensation is suspended and MTTCAN0TTIR.GTE is set and TTOSC.QCS is reset, or in case of the Disc_Bit = '1', MTTCAN0TTIR.GTD is set.

TUR configuration examples are shown in **Table 37.145** below.

**Table 37.145 TUR Configuration Examples**

| TUR | 8 | 10 | 24 | 50 | 510 | 125000 | 32.5 | 100/12 | 529/17 |
|---|---|---|---|---|---|---|---|---|---|
| NC | $1FFF8_H$ | $1FFFE_H$ | $1FFF8_H$ | $1FFEA_H$ | $1FFFE_H$ | $1E848_H$ | $1FFE0_H$ | $19000_H$ | $10880_H$ |
| MTTCAN0TURCF.DC | $3FFF_H$ | $3333_H$ | $1555_H$ | $0A3D_H$ | $0101_H$ | $0001_H$ | $0FC0_H$ | $3000_H$ | $0880_H$ |

MTTCAN0TTOCN.ECS schedules NC for activation by the next reference message. MTTCAN0TTOCN.SGT schedules MTTCAN0TTGTP.TP for activation by the next reference message. Setting of MTTCAN0TTOCN.ECS and MTTCAN0TTOCN.SGT requires MTTCAN0TTOCF.EECS to be set (external clock synchronization enabled) while the M_TTCAN is actual time master.

The M_TTCAN module provides an application watchdog to verity the function of the application program. The Host has to serve this watchdog regularly, else all CAN bus activity is stopped. The Application Watchdog Limit MTTCAN0TTOCF.AWL specifies the number of NTUs between two times the watchdog has to be served. The maximum number of NTUs is 256. The Application Watchdog is served by reading register MTTCAN0TTOST. MTTCAN0TTOST.AWE indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, interrupt flag MTTCAN0TTIR.AW is set. For software development, the application watchdog may be disabled by programming MTTCAN0TTOCF.AWL to $00_H$ (see also **(10) Application Watchdog**).

Bosch CAN FD Specification V1.0

(a) Timing of Interface Signals

The timing events which cause a pulse at output m_ttcan_tmp (MTTCAN0TMP) and m_ttcan_rtp (MTTCAN0RTP) are generated in the CAN clock domain. There is a clock domain crossing delay to be considered before the same event is visible in the Host clock domain (MTTCAN0TTIR.TTMI resp. MTTCAN0TTIR.RTMI set). The signals can be connected e.g. to the timing input(s) of another TTCAN node (m_ttcan_swt:MTTCAN0SWT / m_ttcan_evt: MTTCAN0EVT), in order to automatically synchronize two TTCAN networks.

Output m_ttcan_soc (MTTCAN0SOC) gets active whenever a reference message is completed (either transmitted or received). The output is controlled in the Host clock domain.

**(2) Message Scheduling**

MTTCAN0TTOCF.TM controls whether the M_TTCAN operates as a potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message's identifier MTTCAN0TTRMC.RID define the master priority, 0 giving the highest and 7 giving the lowest priority. There may not be two nodes in the network using the same master priority. MTTCAN0TTRMC.RID is used for recognition of reference messages. MTTCAN0TTRMC.RMPS is not relevant for time slaves.

The Initial Reference Trigger Offset MTTCAN0TTOCF.IRTO is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for

MTTCAN0TTOCF.IRTO is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, should correspond to their master priority, even with maximum clock drift.

MTTCAN0TTOCF.OM decides whether the node operates in TTCAN Level 0, Level 1, or Level 2. In one network, all potential time masters have to operate on the same level. Time slaves may operate on Level 1 in a Level 2 network, but not vice versa. The configuration of the TTCAN operation mode via MTTCAN0TTOCF.OM is the last step in the setup. With MTTCAN0TTOCF.OM = “00” (event-driven CAN communication), the M_TTCAN operates according to ISO 11898-1, without time triggers. With MTTCAN0TTOCF.OM = “01” (Level 1), the M_TTCAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the Next_is_Gap bit in the reference message is ignored. With MTTCAN0TTOCF.OM = “10” (Level 2), the M_TTCAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With MTTCAN0TTOCF.OM = “11” (Level 0), the M_TTCAN operates as event-driven CAN but maintains a calibrated global time base as in Level 2.

MTTCAN0TTOCF.EECS enables the external clock synchronization, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in Level 0,2 only) the global clock phase to an external reference.

MTTCAN0TTMLM.ENTT in the TT Matrix Limits register specifies the number of expected Tx_Triggers in the system matrix. This is the sum of Tx_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx_Ref_Triggers. Note that this is usually not the number of Tx_Trigger memory elements; the number of basic cycles in the system matrix and the trigger’s repeat factors have to be taken into account. An inaccurate configuration of MTTCAN0TTMLM.ENTT will result in either a Tx Count Underflow (MTTCAN0TTIR.TXU = ‘1’ and MTTCAN0TTOST.EL = “01”, severity 1) or in a Tx Count Overflow (MTTCAN0TTIR.TXO = ‘1’ and MTTCAN0TTOST.EL = “10”, severity 2).

Bosch CAN-FD Specification V1.0

**NOTE**

In case the first reference message seen by a node does not have Cycle_Count zero, this node may finish its first matrix cycle with its Tx count resulting in a Tx Count Underflow condition. As long as a node is in state Synchronizing its Tx_Triggers will not lead to transmissions.

MTTCAN0TTMLM.CCM specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of 8 basic cycles MTTCAN0TTMLM.CCM would be 7. MTTCAN0TTMLM.CCM is ignored by time slaves, a receiver of a reference message considers the received cycle count as the valid cycle count for the actual basic cycle.

MTTCAN0TTMLM.TXEW specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If the sample point of the first bit of a transmit message is not inside the Tx enable window because of e.g. a slight overlap from the previous time window’s message, the transmission cannot be started in that time window at all. MTTCAN0TTMLM.TXEW has to be chosen with respect to the network’s synchronization quality and with respect to the relation between the length of the time windows and the length of the messages.

### (3) Trigger Memory

The trigger memory is part of the external Message RAM to which the M_TTCAN is connected via its Generic Master Interface (see **Figure 37.15, Message RAM Configuration**). It stores up to 64 trigger elements. A trigger memory element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, Filter Type FTYPE, Message Number MNR, Message Status Count MSC, Time Mark Event Internal TMIN, Time Mark Event External TMEX, and Asynchronous Serial Communication ASC (see **(7) Trigger Memory Element**).

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory have to be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are ignored for triggers of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, and End_of_List.

When the cycle time reaches the time mark of the actual trigger, the FSE switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a transmit trigger, the Tx Handler starts to read the message from the Message RAM as soon as the FSE switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx Handler has to be able to prepare the transmission before the transmit trigger's time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mark is reached. If the reference message is n NTU long, a trigger with a time mark< n will never become active and will be treated as a configuration error.

Starting point of the cycle time is the sample point of the reference message's start of frame bit. The next reference message is requested when cycle time reaches the Tx_Ref_Trigger's time mark. The M_TTCAN reacts on the transmission request at the next sample point. A new Sync_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync_Mark is taken as the new Ref_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialisation) be seen at a value< n, with n being the length of the reference message measured in NTU.

Length of a basic cycle: Tx_Ref_Trigger's time mark+1 NTU+1 CAN bit time

The trigger list will be different for all nodes in the TTCAN network. Each node knows only the Tx_Triggers for its own transmit messages, the Rx_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

#### (a) Trigger Types

Tx_Ref_Trigger (TYPE = "0000") and Tx_Ref_Trigger_Gap (TYPE = "0001") cause the transmission of a reference message by a time master. A configuration error (MTTCAN0TTOST.EL = "11", severity 3) is detected when a time slave encounters a Tx_Ref_Trigger(_Gap) in its trigger memory. Tx_Ref_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, Tx_Ref_Trigger is ignored when the M_TTCAN synchronization state is In_Gap (MTTCAN0TTOST.SYS = "10").

Tx_Trigger_Single (TYPE = "0010"), Tx_Trigger_Continuous (TYPE = "0011"), Tx_Trigger_Arbitration (TYPE = "0100"), and Tx_Trigger_Merged (TYPE = "0101") cause the start of a transmission. They define the start of a time window.

Tx_Trigger_Single starts a single transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit is reset.

Bosch CAN-FD Specification V1.0

Tx_Trigger_Continuous starts a transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit remains set, and the message buffer is transmitted again in the next matching time window.

Tx_Trigger_Arbitration starts an arbitrating time window, Tx_Trigger_Merged a merged arbitrating time window. The last Tx_Trigger of a merged arbitrating time window must be of type Tx_Trigger_Arbitration. A Configuration Error (MTTCAN0TTOST.EL = "11", severity 3) is detected when a trigger of type Tx_Trigger_Merged is followed by any other Tx_Trigger than one of type Tx_Trigger_Merged or Tx_Trigger_Arbitration. Several Tx_Triggers may be defined for the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code has to be considered when the expected number of Tx_Triggers (MTTCAN0TTMLM.ENTT) is calculated.

Watch_Trigger (TYPE = "0110") and Watch_Trigger_Gap (TYPE = "0111") check for missing reference messages. They are used by both time masters and time slaves. Watch_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch_Trigger is ignored when the M_TTCAN synchronization state is In_Gap (MTTCAN0TTOST.SYS = "10").

Rx_Trigger (TYPE = "1000") is used to check for the reception of periodic messages in exclusive time windows. Rx_Triggers are not active until state In_Schedule or In_Gap is reached. The time mark of an Rx_Trigger shall be placed after the end of that message's transmission, independent of time window boundaries. Depending on their cycle code, Rx_Triggers may be ignored in some basic cycles. At the time mark of the Rx_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx_Trigger had matched the acceptance filter element referenced by MNR. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx_Trigger. Acceptance filter elements which are referenced by Rx_Triggers should be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx_Trigger's time mark is reached.

Bosch CAN-FD Specification V1.0

Time_Base_Trigger (TYPE = "1001") are used to generate internal/external events depending on the configuration of ASC, TMIN, and TMEX.

End_of_List (TYPE = "1010 to 1111") is an illegal trigger type, a configuration error (MTTCAN0TTOST.EL = "11", severity 3) is detected when an End_of_List trigger is encountered in the trigger memory before the Watch_Trigger or Watch_Trigger_Gap.

(b) Restrictions for the Node's Trigger List

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark.

Rx_Triggers and Time_Base_Triggers may not be placed inside the Tx enable windows of Tx_Trigger_Single/Continuous/Arbitration, but they may be placed after Tx_Trigger_Merged.

Triggers that are placed after the Watch_Trigger (or the Watch_Trigger_Gap when MTTCAN0TTOST.SYS = "10") will never become active. The watch triggers themselves will not become active when the reference messages are transmitted on time.

All unused trigger memory words (after the Watch_Trigger or after the Watch_Trigger_Gap when MTTCAN0TTOST.SYS = "10") must be set to trigger type End_of_List.

A typical trigger list for a potential time master will begin with a number of Tx_Triggers and Rx_Triggers followed by the Tx_Ref_Trigger and the Watch_Trigger. For networks with external event- synchronized time-triggered communication, this is followed by the Tx_Ref_Trigger_Gap and the Watch_Trigger_Gap. The trigger list for a time slave will be the same but without the Tx_Ref_Trigger and the Tx_Ref_Trigger_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger's time mark and activates the trigger. Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx_Ref_Trigger and Tx_Ref_Trigger_Gap. In a time master competing for master ship, the effective time mark of a Tx_Ref_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx_Ref_Trigger or Tx_Ref_Trigger_Gap is the sum of its configured time mark and the Reference Trigger Offset MTTCAN0TTOCF.IRTO. In case error level 2 is reached (MTTCAN0TTOST.EL = "10"), the effective time mark is the sum of its time mark and $127_H$. No other trigger elements should be placed in this range otherwise it may happen, that the time marks appear out of order and are flagged as a configuration error. Trigger elements which are coming after Tx_Ref_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:

- Host clock frequency
- Speed and waiting time for Trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

### (c) Example for Trigger Handling

The example below shows how the trigger list is derived from a node's system matrix. Assumed node A is first time master and has knowledge of the section of the system matrix shown in **Table 37.146** below.

Bosch CAN-FD Specification V1.0

**Table 37.146 System Matrix Node A**

| Cycle Count | Time Mark1 | Time Mark2 | Time Mark3 | Time Mark4 | Time Mark5 | Time Mark6 | Time Mark7 |
|---|---|---|---|---|---|---|---|
| 0 | Tx7 | | | | | TxRef | Error |
| 1 | Rx3 | | | Tx2, Tx4 | | TxRef | Error |
| 2 | | | | | | TxRef | Error |
| 3 | Tx7 | | Rx5 | | | TxRef | Error |
| 4 | Tx7 | | | Rx6 | | TxRef | Error |

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the number of basic cycles in the system matrix is 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by MTTCAN0TTMLM.CCM. The Cycle Code CC is composed of repeat factor (= value of most significant '1') and the number of the first basic cycle in the system matrix (= bit field after most significant '1').

Example:with a cycle code of 0b0010011 (repeat factor: 16, first basic cycle: 3) and a maximum cycle count of MTTCAN0TTMLM.CCM = "$3F_H$"matches occur at cycle counts 3, 19, 35, 51

A trigger element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, and Message Number MNR. For transmission MNR references the Tx Buffer number (0 to 31). For reception MNR references the number of the filter element (0 to 127) that matched during acceptance filtering.

Depending on the configuration of the Filter Type FTYPE, the 11-bit or 29-bit message ID filter list is referenced.

In addition a trigger element can be configured for Asynchronous Serial Communication ASC, generation of Time Mark Event Internal TMIN, and Time Mark Event External TMEX. The Message Status Count MSC holds the counter value (0 to 7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

**Table 37.147 Trigger List Node A**

| Trigger | Time Mark TM[15:0] | Cycle Code CC[6:0] | Trigger Type TYPE[3:0] | Mess. No. MNR[6:0] |
|---|---|---|---|---|
| 0 | Mark1 | $0000100_B$ | Tx_Trigger_Single | 7 |
| 1 | Mark1 | $1000000_B$ | Rx_Trigger | 3 |
| 2 | Mark1 | $1000011_B$ | Tx_Trigger_Single | 7 |
| 3 | Mark3 | $1000001_B$ | Tx_Trigger_Merged | 2 |
| 4 | Mark3 | $1000011_B$ | Rx_Trigger | 5 |
| 5 | Mark4 | $1000001_B$ | Tx_Trigger_Arbitration | 4 |
| 6 | Mark4 | $1000100_B$ | Rx_Trigger | 6 |
| 7 | Mark6 | n.a. | Tx_Ref_Trigger | 0 (Ref) |
| 8 | Mark7 | n.a. | Watch_Trigger | n.a. |
| 9 | n.a. | n.a. | End_of_List | n.a. |

Tx_Trigger_Single, Tx_Trigger_Continuous, Tx_Trigger_Merged, Tx_Trigger_Arbitration, Rx_Trigger, and Time_Base_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored.

The FSE starts the basic cycle with scanning the trigger list starting from zero until a trigger with time mark > cycle time and with its Cycle Code CC matching the actual cycle count is reached, or a trigger of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, or Watch_Trigger_Gap is encountered.

When the cycle time reached the Time Mark TM, the action defined by Trigger Type TYPE and Message Number MNR is started. There is an error in the configuration when End_of_List is reached.

At Mark6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the Watch Trigger at Mark7 is reached, the node was not able to transmit the reference message; error treatment is started.

Bosch CAN-FD Specification V1.0

### (d) Detection of Configuration Errors

A configuration error is signalled via MTTCAN0TTOST.EL = “11” (severity 3) when:

The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time.

The previous active trigger was a Tx_Trigger_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a Tx_Trigger_Merged nor a Tx_Trigger_Arbitration nor a Time_Base_Trigger nor an Rx_Trigger.

The FSE of a node with MTTCAN0TTOCF.TM=‘0’ (time slave) encounters a Tx_Ref_Trigger or a Tx_Ref_Trigger_Gap.

Any time mark placed inside the Tx enable window (defined by MTTCAN0TTMLM.TXEW) of a Tx_Trigger with a matching cycle code.

A time mark is placed near the time mark of a Tx_Ref_Trigger and the Reference Trigger Offset MTTCAN0TTOST.RTO causes a reversal of their sequential order measured in cycle time.

### (4) TTCAN Schedule Initialization

The synchronization to the M_TTCAN's message schedule starts when MTTCAN0CCCR.INIT is reset. The M_TTCAN can operate strictly time-triggered (MTTCAN0TTOCF.GEN = '0') or external event-synchronized time-triggered (MTTCAN0TTOCF.GEN = '1'). All nodes start with cycle time zero at the beginning of their trigger list with MTTCAN0TTOST.SYS = "00" (out of synchronization), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronized time-triggered operation mode will ignore Tx_Ref_Trigger and Watch_Trigger and will use instead Tx_Ref_Trigger_Gap and Watch_Trigger_Gap until the first reference message decides whether a Gap is active.

#### (a) Time Slaves

After configuration, a time slave will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message before reaching the Watch_Triggers. When it reaches Init_Watch_Trigger, interrupt flag MTTCAN0TTIR.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). The first received reference message will restart the FSE and the cycle time.

**NOTE**

Init_Watch_Trigger is not part of the trigger list. It is implemented as an internal counter which counts up to $FFFF_H$ = maximum cycle time.

When a time slave has received any message but the reference message before reaching the Watch_Triggers, it will assume a fatal error (MTTCAN0TTOST.EL = "11", severity 3), set interrupt flag MTTCAN0TTIR.WT, switch off its CAN bus output, and enter the bus monitoring mode (MTTCAN0CCCR.MON set to '1'). In the bus monitoring mode it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

**NOTE**

To leave the fatal error state, the Host has to set MTTCAN0CCCR.INIT = '1'. After reset of MTTCAN0CCCR.INIT, the node restarts TTCAN communication.

When no error is encountered during synchronization, the first reference message sets MTTCAN0TTOST.SYS = "01" (Synchronizing), the second sets the TTCAN synchronization state (depending on its Next_is_Gap bit) to MTTCAN0TTOST.SYS = "11" (In_Schedule) or MTTCAN0TTOST.SYS = "10" (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

#### (b) Potential Time Masters

After configuration, a potential time master will start the transmission of a reference message when it reaches its Tx_Ref_Trigger (or its Tx_Ref_Trigger_Gap when in external event-synchronized time-triggered operation). It will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message or transmit the reference message successfully before reaching the Watch_Triggers (assumed reason: all other nodes still in reset or configuration, giving no acknowledge). When it reaches Init_Watch_Trigger, the attempted transmission is aborted, interrupt flag MTTCAN0TTIR.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting MTTCAN0TTIR.IWT will re-enable the transmission of reference messages until next time the Init_Watch_Trigger condition is met, or another CAN message is received. The FSE will be restarted by the reception of a reference message.

When a potential time master reaches the Watch_Triggers after it has received any message but the reference message, it will assume a fatal error (MTTCAN0TTOST.EL = “11”, severity 3), set interrupt flag MTTCAN0TTIR.WT, switch off its CAN bus output, and enter the bus monitoring mode (MTTCAN0CCCR.MON set to ‘1’). In bus monitoring mode, it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets MTTCAN0TTOST.SYS = “01” (synchronizing), the second sets the TTCAN synchronization state (depending on its Next_is_Gap bit) to MTTCAN0TTOST.SYS = “11” (In_Schedule) or MTTCAN0TTOST.SYS = “10” (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

A potential time master is current time master (MTTCAN0TTOST.MS = “11”) when it was the transmitter of the last reference message, else it is backup time master (MTTCAN0TTOST.MS = “10”).

When all potential time masters have finished configuration, the node with the highest time master priority in the network will become the current time master.

### 37.6.4.3 TTCAN Gap Control

All functions related to Gap control apply only when the M_TTCAN is operated in external event-synchronized time-triggered mode (MTTCAN0TTOCF.GEN = ‘1’). In this operation mode the TTCAN message schedule may be interrupted by inserting Gaps between the basic cycles of the system matrix. All nodes connected to the CAN network have to be configured for external event-synchronized time-triggered operation.

During a Gap, all transmissions are stopped and the CAN bus remains idle. A Gap is finished when the next reference message starts a new basic cycle. A Gap starts at the end of a basic cycle that itself was started by a reference message with bit Next_is_Gap = ‘1’ e.g. Gaps are initiated by the current time master.

Bosch CAN FD Specification V1.0

The current time master has two options to initiate a Gap. A Gap can be initiated under software control when the application program writes MTTCAN0TTOCN.NIG = ‘1’. The Next_is_Gap bit will be transmitted as ‘1’ with the next reference message. A Gap can also be initiated under hardware control when the application program enables the event trigger input pin m_ttcan_evt (MTTCAN0EVT) by writing MTTCAN0TTOCN.GCS = ‘1’. When a reference message is started and MTTCAN0TTOCN.GCS is set, a HIGH level at pin m_ttcan_evt (MTTCAN0EVT) will set Next_is_Gap = ‘1’.

As soon as that reference message is completed, the MTTCAN0TTOST.WFE bit will announce the Gap to the time master as well as to the time slaves. The current basic cycle will continue until its last time window. The time after the last time window is the Gap time.

For the actual time master and the potential time masters, MTTCAN0TTOST.GSI will be set when the last basic cycle has finished and the Gap time starts. In nodes that are time slaves, bit MTTCAN0TTOST.GSI will remain at ‘0’.

When a potential time master is in synchronization state In_Gap (MTTCAN0TTOST.SYS = “10”), it has four options to intentionally finish a Gap:

Under software control by writing MTTCAN0TTOCN.FGP = ‘1’.

Under hardware control (MTTCAN0TTOCN.GCS = ‘1’) an edge from HIGH to LOW at the event-trigger input pin m_ttcan_evt (MTTCAN0EVT) sets MTTCAN0TTOCN.FGP and restarts the schedule.

The third option is a time-triggered restart. When MTTCAN0TTOCN.TMG = '1', the next register time mark interrupt (MTTCAN0TTIR.RTMI = '1') will set MTTCAN0TTOCN.FGP and start the reference message.

Finally any potential time master will finish a Gap when it reaches its Tx_Ref_Trigger_Gap, assuming that the event to synchronize on did not occur in time.

Neither of these options can cause a basic cycle to be interrupted with a reference message.

Setting of MTTCAN0TTOCN.FGP after the Gap time has started will start the transmission of a reference message immediately and will thereby synchronize the message schedule. When MTTCAN0TTOCN.FGP is set before the Gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the Tx_Ref_Trigger – there will be no Gap time in the message schedule.

In strictly time-triggered operation, bit Next_is_Gap = '1' in the reference message will be ignored, as well as the event-trigger input pin m_ttcan_evt (MTTCAN0EVT) and the bits MTTCAN0TTOCN.NIG, MTTCAN0TTOCN.FGP, and MTTCAN0TTOCN.TMG.

#### 37.6.4.4 Stop Watch

The stop watch function enables capturing of M_TTCAN internal time values (local time, cycle time, or global time) triggered by an external event.

To enable the stop watch function, the application program first has to define local time, cycle time, or global time as stop watch source via MTTCAN0TTOCN.SWS. When MTTCAN0TTOCN.SWS is ≠ "00" and TT Interrupt Register flag MTTCAN0TTIR.SWE is '0', the actual value of the time selected by MTTCAN0TTOCN.SWS will be copied into MTTCAN0TTCPT.SWV on the next rising/falling edge (as configured via MTTCAN0TTOCN.SWP) on pin m_ttcan_swt (MTTCAN0SWT). This will set interrupt flag MTTCAN0TTIR.SWE. After the application program has read MTTCAN0TTCPT.SWV, it may enable the next stop watch event by resetting MTTCAN0TTIR.SWE to '0'.

Bosch CAN FD Specification V1.0

#### 37.6.4.5 Local Time, Cycle Time, Global Time, and External Clock Synchronization

There are two possible levels in time-triggered CAN: Level 1 and Level 2. Level 1 only provides time-triggered operation using cycle time. Level 2 additionally provides increased synchronization quality, global time and external clock synchronization. In both levels, all timing features are based on a local time base - the local time.

The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented 8 times each NTU. If the length of the NTU is shorter than 8 CAN clock periods (as may be configured in Level 1, or as a result of clock calibration in Level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only 4 times each NTU.

**Figure 37.25** describes the synchronization of the cycle time and global time, performed in the same manner by all TTCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message's frame synchronization event. This frame synchronization event occurs at the sample point of each Start of Frame (SoF) bit and causes the local time to be stored as Sync_Mark. Sync_Marks and Ref_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the Cycle Sync Mark

(Cycle Sync Mark = Sync_Mark - Ref_Mark) stored in register MTTCAN0TTCSM. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (Cycle Time = Local Time - Ref_Mark).

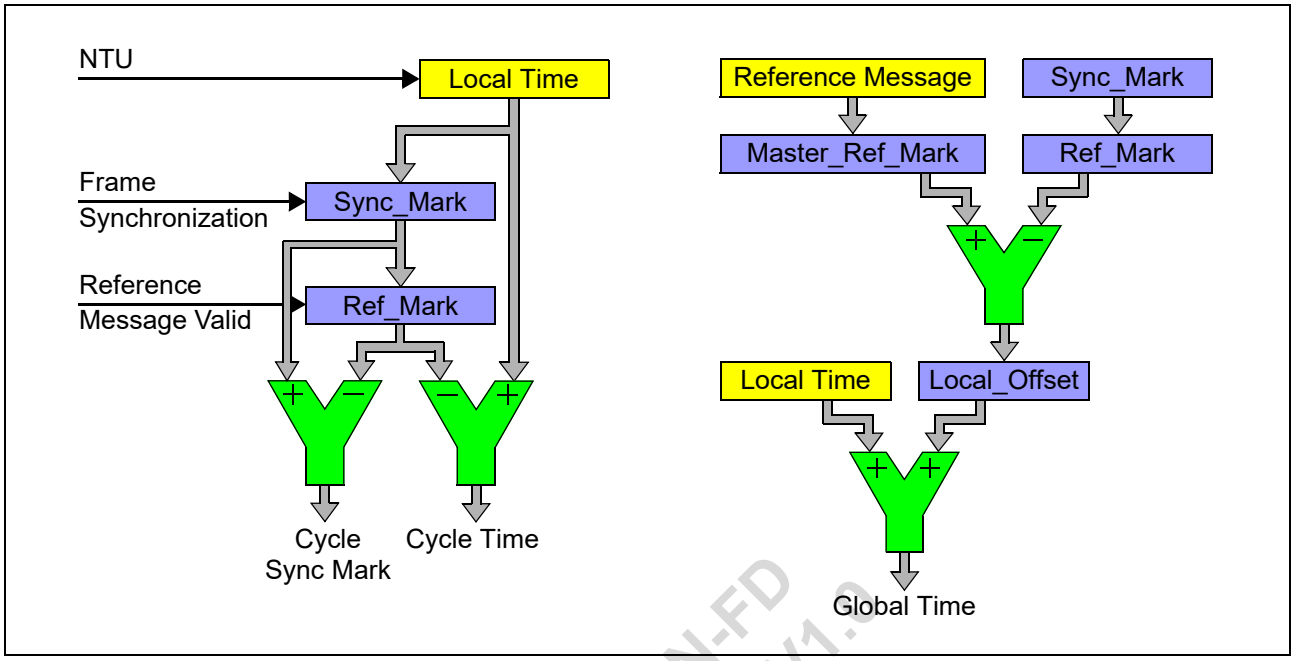



**Figure 37.25 Cycle Time and Global Time Synchronization**

Bosch CAN-FD Specification V1.0

The cycle time that can be read from MTTCAN0TTCTC.CT is the difference of the node's local time and Ref_Mark, both synchronized into the Host clock domain and truncated to 16 bit.

The global time exists for TTCAN Level 0 and Level 2 only, in Level 1 it is invalid. The node's view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master will use its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3,4). The global time that can be read from MTTCAN0TTLGT.GT is the sum of the node's local time and its local offset, both synchronized into the Host clock domain and truncated to 16 bit. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark (see **Figure 37.25**). The node's view of the global time is local time + local offset. In a potential time master that has never received another time master's reference message, Local_Offset will be zero. When a node becomes the current time master after first having received other reference messages, Local_Offset will be frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signalled by Disc_Bit in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register MTTCAN0TTLGT is smoothed by a low-pass filtering to have a continuous monotonic value.

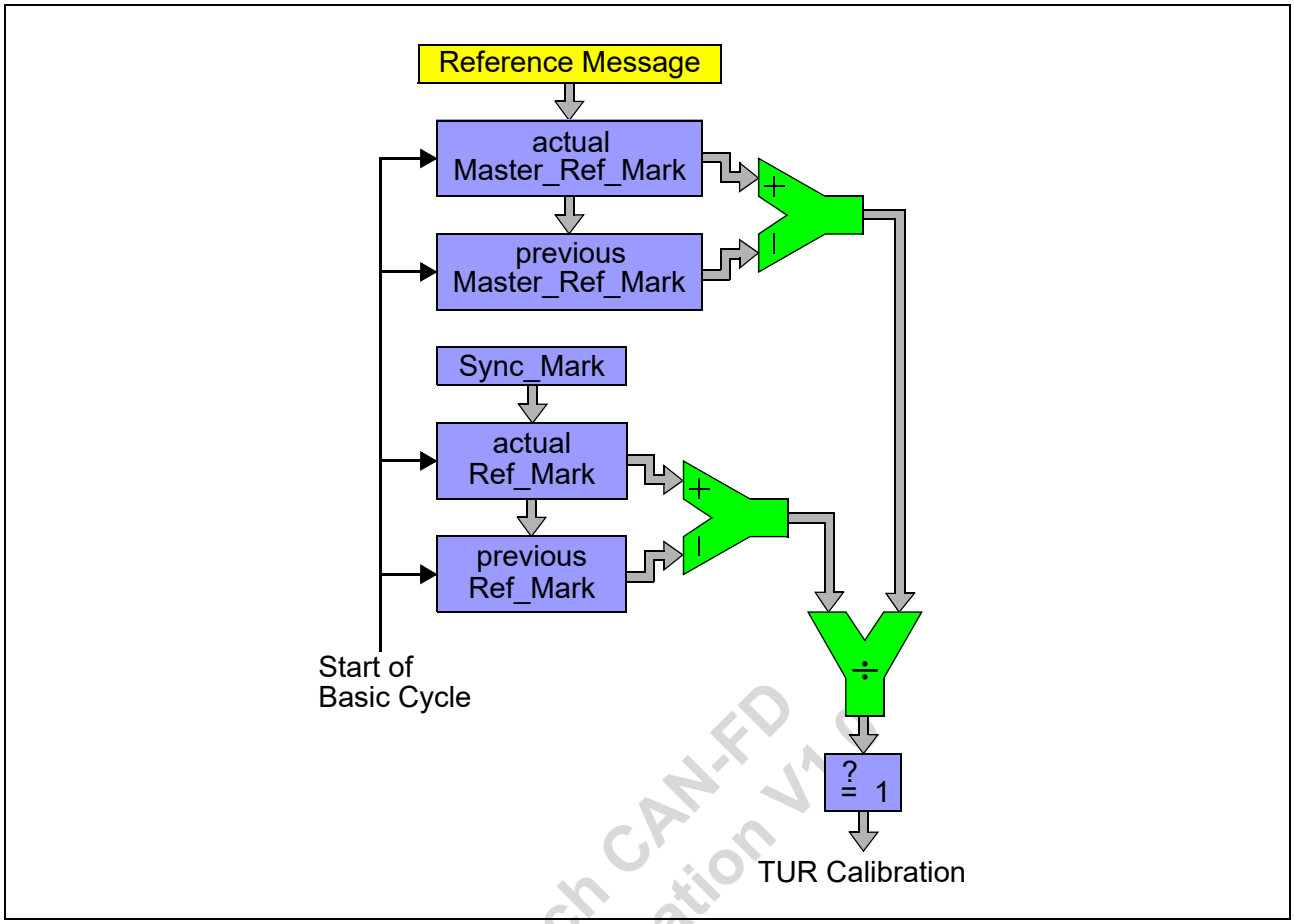



Figure 37.26 TTCAN Level 0 and Level 2 Drift Compensation

**Figure 37.26** describes how in TTCAN Level 0,2 each time receiving node compensates the drift between its own local clock and the time master's clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the Disc_Bit in the reference message is not set, a new value for MTTCAN0TURNA.NAV is calculated. If the Synchronization Deviation SD = |NC - MTTCAN0TURNA.NAV| ≤ SDL (Synchronization Deviation Limit), the new value for MTTCAN0TURNA.NAV takes effect. Else the automatic drift compensation is suspended.

In TTCAN Level 0 and Level 2, MTTCAN0TTOST.QCS indicates whether the automatic drift compensation is active or suspended. In TTCAN Level 1, MTTCAN0TTOST.QCS is always '1'.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit MTTCAN0TTOCF.EECS.

The stop watch function (see **Section 37.6.4.4, Stop Watch**) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated Numerator Configuration Low to MTTCAN0TURCF.NCL (MTTCAN0TURCF.DC cannot be updated during operation). The new value takes effect by writing MTTCAN0TTOCN.ECS to '1'.

The global time phase is adjusted by first writing the phase offset into the TT Global Time Preset register MTTCAN0TTGTP. The new value takes effect by writing MTTCAN0TTOCN.SGT to '1'. The first reference message transmitted after the global time phase adjustment will have the Disc_Bit set to '1'.

Bosch CAN-FD Specification V1.0

MTTCAN0TTOST.QGTP shows whether the node's global time is in phase with the time master's global time. MTTCAN0TTOST.QGTP is permanently '0' in TTCAN Level 1 and when the Synchronization Deviation Limit is exceeded in TTCAN Level 0,2 (MTTCAN0TTOST.QCS = '0'). It is temporarily '0' while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a Disc_Bit = '1' or when MTTCAN0TTOST.QCS = '0'.

### 37.6.4.6 TTCAN Error Level

The ISO 11898-4 specifies four levels of error severity:

#### S0 - No Error

#### S1 - Warning

Only notification of application, reaction application-specific.

#### S2 Error

Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the Reference Trigger Offset MTTCAN0TTOST.RTO set to the maximum value of 127.

#### S3 - Severe Error

Notification of application. All CAN bus operations are stopped, i.e. transmission of dominant bits is not allowed, and MTTCAN0CCCR.MON is set. The S3 error condition remains active until the application updates the configuration (set MTTCAN0CCCR.CCE).

If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by MTTCAN0TTIR.ELC. The error level is monitored by MTTCAN0TTOST.EL.

The M_TTCAN signals the following error conditions as required by ISO 11898-4:

#### Config_Error (S3)

Sets Error Level MTTCAN0TTOST.EL to "11" when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.

#### Watch_Trigger_Reached (S3)

Sets Error Level MTTCAN0TTOST.EL to "11" when a watch trigger was reached because the reference message is missing.

#### Application_Watchdog (S3)

Sets Error Level MTTCAN0TTOST.EL to "11" when the application failed to serve the application watchdog. The application watchdog is configured via MTTCAN0TTOCF.AWL. It is served by reading register MTTCAN0TTOST. When the watchdog is not served in time, bit MTTCAN0TTOST.AWE and interrupt flag MTTCAN0TTIR.AW are set, all TTCAN communication is stopped, and the M_TTCAN is set into bus monitoring mode (MTTCAN0CCCR.MON set to '1').

Bosch CAN-FD Specification V1.0

#### CAN_Bus_Off (S3)

Entering CAN_Bus_Off state sets error level MTTCAN0TTOST.EL to “11”. CAN_Bus_Off state is signalled by MTTCAN0PSR.BO = ‘1’ and MTTCAN0CCCR.INIT = ‘1’.

#### Scheduling_Error_2 (S2)

Sets Error Level MTTCAN0TTOST.EL to “10” if the MSC of one Tx_Trigger has reached 7. In addition interrupt flag MTTCAN0TTIR.SE2 is set. The Error Level MTTCAN0TTOST.EL is reset to “00” at the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.

#### Tx_Overflow (S2)

Sets Error Level MTTCAN0TTOST.EL to “10” when the Tx count is equal or higher than the expected number of Tx_Triggers MTTCAN0TTMLM.ENTT and a Tx_Trigger event occurs. In addition interrupt flag MTTCAN0TTIR.TXO is set. The Error Level MTTCAN0TTOST.EL is reset to “00” when the Tx count is no more than MTTCAN0TTMLM.ENTT at the start of a new matrix cycle.

#### Scheduling_Error_1 (S1)

Sets Error Level MTTCAN0TTOST.EL to “01” if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than 2, or if one of the MSCs of an exclusive Rx_Trigger has reached 7. In addition interrupt flag MTTCAN0TTIR.SE1 is set. If within one matrix cycle none of these conditions is valid, the Error Level MTTCAN0TTOST.EL is reset to “00”.

#### Tx_Underflow (S1)

Sets Error Level MTTCAN0TTOST.EL to “01” when the Tx count is less than the expected number of Tx_Triggers MTTCAN0TTMLM.ENTT at the start of a new matrix cycle. In addition interrupt flag MTTCAN0TTIR.TXU is set. The Error Level MTTCAN0TTOST.EL is reset to “00” when the Tx count is at least MTTCAN0TTMLM.ENTT at the start of a new matrix cycle.

Bosch CAN FD Specification V1.0

### 37.6.4.7 TTCAN Message Handling

#### (1) Reference Message

For potential time masters the identifier of the reference message is configured via MTTCAN0TTRMC.RID. No dedicated Tx Buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte (TTCAN Level 1) resp. the first four data bytes (TTCAN Level 0 and Level 2) will be provided by the FSE.

In case the reference message Payload Select MTTCAN0TTRMC.RMPS is set, the rest of the reference message's payload (Level 1: bytes 2-8, Level 0,2: bytes 5-6) is taken from Tx Buffer 0. In this case the data length DLC code from message buffer 0 is used.

**Table 37.148 Number of Data Bytes transmitted with a reference messages**

| MTTCAN0TTRMC.RMPS | MTTCAN0TXBRP.TRP0 | Level 0 | Level 1 | Level 2 |
|---|---|---|---|---|
| 0 | 0 | 4 | 1 | 4 |
| 0 | 1 | 4 | 1 | 4 |
| 1 | 0 | 4 | 1 | 4 |
| 1 | 1 | 4 + MB0 | 1 + MB0 | 4 + MB0 |

To send additional payload with the reference message in Level 1 a DLC > 1 has to be configured, for Level 0,2 a DLC > 4 is required. In addition the transmission request pending bit MTTCAN0TXBRP.TRP0 of message buffer 0 must be set (see **Table 37.148**). In case bit MTTCAN0TXBRP.TRP0 is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the Reference Identifier MTTCAN0TTRMC.RID is used.

Bosch CAN-FD Specification V1.0

#### (2) Message Reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see **Section 37.6.3.4, Rx Handling**).

The Message Status Count MSC is part of the corresponding trigger memory element and has to be initialized to zero during configuration. It is updated while the M_TTCAN is in synchronization states In_Gap or In_Schedule. The update happens at the message's Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has completed. This has to take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

### (3) Message Transmission

For time-triggered message transmission the M_TTCAN supplies 32 dedicated Tx buffers (see **Section (2), Dedicated Tx Buffers**). A Tx FIFO or Tx queue is not available when the M_TTCAN is configured for time-triggered operation (MTTCAN0TTOCF.OM = “01” or “10”).

Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle.

The application program has to update the data regularly and on time, synchronized to the cycle time. The Host CPU is responsible that no partially updated messages are transmitted. To assure this the Host has to proceed in the following way:

Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration

- Check whether the previous transmission has completed by reading MTTCAN0TXBTO
- Update the Tx buffer’s configuration and/or payload
- Issue an Add Request to set the Tx Buffer Request Pending bit

Tx_Trigger_Continuous

- Issue a Cancellation Request to reset the Tx Buffer Request Pending bit
- Check whether the cancellation has finished by reading MTTCAN0TXBCF
- Update Tx buffer’s configuration and/or payload
- Issue an Add Request to set the Tx Buffer Request Pending bit

Bosch CAN-FD Specification V1.0

The message’s MSC stored with the corresponding Tx_Trigger provides information on the success of the transmission.

The MSC is incremented by one when the transmission could not be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and could not be completed successfully. The MSC is decremented by one when the message was transmitted successfully or when the message could have been started within its transmit enable window but was not started because transmission was disabled (M_TTCAN in Error Level S2 or Host has disabled this particular message).

The Tx buffers may be managed dynamically, i.e. several messages with different identifiers may share the same Tx buffer element. In this case the Host has to assure that no transmission request is pending for the Tx buffer element to be reconfigured by checking MTTCAN0TXBRP.

If a Tx buffer with pending transmission request should be updated, the Host first has to issue a cancellation request and check whether the cancellation has completed by reading MTTCAN0TXBCF before it starts updating.

The Tx Handler will transfer a message from the Message RAM to its intermediate output buffer at the trigger element which becomes active immediately before the Tx_Trigger element which defines the beginning of the transmit window. During and after the transfer time the transmit message may not be updated and its MTTCAN0TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be e.g. a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger has to be large enough to guarantee that the Tx Handler can read four words from the Message RAM even at high RAM access load from other modules.

(a) Transmission in Exclusive Time Windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continuous. There is no arbitration on the bus with messages from other nodes. The MSC is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx Buffer Request Pending bit is reset. After successful transmission started by a Tx_Trigger_Continuous the respective Tx Buffer Request Pending remains set. When the transmission was not successful due to disturbances, it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

(b) Transmission in Arbitrating Time Windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message has to arbitrate with the messages from other nodes. The MSC is not updated. When the transmission was not successful (lost arbitration or disturbance), it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

(c) Transmission in Merged Arbitrating Time Windows

The purpose of a merged arbitrating time window is to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a single node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission will not be started if the corresponding Transmission Request Pending flag was reset by a successful Tx cancellation.

In single transmit windows, the Tx Handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmissions will be attempted in the sequence defined by the trigger list. If the time mark of a fourth message is read before the first is transmitted (or cancelled by the Host), the fourth request will be ignored.

The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window will be started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if there is competing bus traffic. This has to be considered for the configuration of the trigger list. Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx Buffer needs to be updated.

Bosch CAN-FD Specification V1.0

#### 37.6.4.8 TTCAN Interrupt and Error Handling

The TT Interrupt Register MTTCAN0TTIR consists of four segments. Each interrupt can be enabled separately by the corresponding bit in the TT Interrupt Enable register MTTCAN0TTIE. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position.

The first segment consists of flags CER, AW, WT, and IWT. Each flag indicates a fatal error condition where the CAN communication is stopped. With the exception of IWT, these error conditions require a re-configuration of the M_TTCAN module before the communication can be restarted.

The second segment consists of flags ELC, SE1, SE2, TXO, TXU, and GTE. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, e.g. by disturbances on the CAN bus, they will be handled by the TTCAN protocol's failure handling and do not require intervention by the application program.

The third segment consists of flags GTD, GTW, SWE, TTMI, and RTMI. The first two flags are controlled by global time events (Level 0, 2 only) that require a reaction by the application program. With a Stop Watch Event triggered by a rising/falling edge on pin m_ttcan_swt (MTTCAN0SWT) internal time values are captured. The Trigger Time Mark Interrupt notifies the application that a specific Time_Base_Trigger is reached. The Register Time Mark Interrupt signals that the time referenced by MTTCAN0TTOCN.TMC (Cycle, Local, or Global) equals time mark MTTCAN0TTTMK.TM. It can also be used to finish a Gap.

The fourth segment consists of flags SOG, CSM, SMC, and SBC. These flags provide a means to synchronize the application program to the communication schedule.

Bosch CAN-FD Specification V1.0

### 37.6.4.9 Level 0

TTCAN Level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in TTCAN Level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1.

Level 0 operation is configured via MTTCAN0TTOCF.OM = “11”. In this mode the M_TTCAN operates in event-driven CAN communication, there is no fixed schedule, the configuration of MTTCAN0TTOCF.GEN is ignored. External event-synchronized operation is not available in Level 0. A synchronized time base is maintained by transmission of reference messages.

In Level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (MTTCAN0TTIR.TTMI) is set when the cycle time has reached MTTCAN0TTOCF.IRTO • $200_H$, it reminds the Host to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (MTTCAN0TTIR.WT) is set when the cycle time has reached $FF00_H$. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks.

Register time mark interrupts (MTTCAN0TTIR.RTMI) are also possible.

The reference message is configured as for Level 2 operation. Received reference messages are recognized by the identifier configured in register MTTCAN0TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the Host sets a transmission request for message buffer 0, there is no reference trigger offset.

Bosch CAN-FD Specification V1.0

Level 0 operation is configured via:

- MTTCAN0TTRMC
- MTTCAN0TTOCF except EVTP, AWL, GEN
- MTTCAN0TTMLM except ENTT, TXEW
- MTTCAN0TURCF

Level 0 operation is controlled via:

- MTTCAN0TTOCN except NIG, TMG, FGP, GCS, TTMIE
- MTTCAN0TTGTP
- MTTCAN0TTTMK
- MTTCAN0TTIR excluding bits CER, AW, IWT SE2, SE1, TXO, TXU, SOG (no function)
- MTTCAN0TTIR the following bits have changed function
  - TTMI not defined by trigger memory - activated at cycle time MTTCAN0TTOCF.IRTO • $200_H$
  - WT not defined by trigger memory - activated at cycle time $FF00_H$

Level 0 operation is signalled via:

- MTTCAN0TTOST excluding bits AWE, WFE, GSI, GFI, RTO (no function)

### (1) Synchronizing

**Figure 37.27** below describes the states and state transitions in TTCAN Level 0 operation. Level 0 has no In_Gap state.

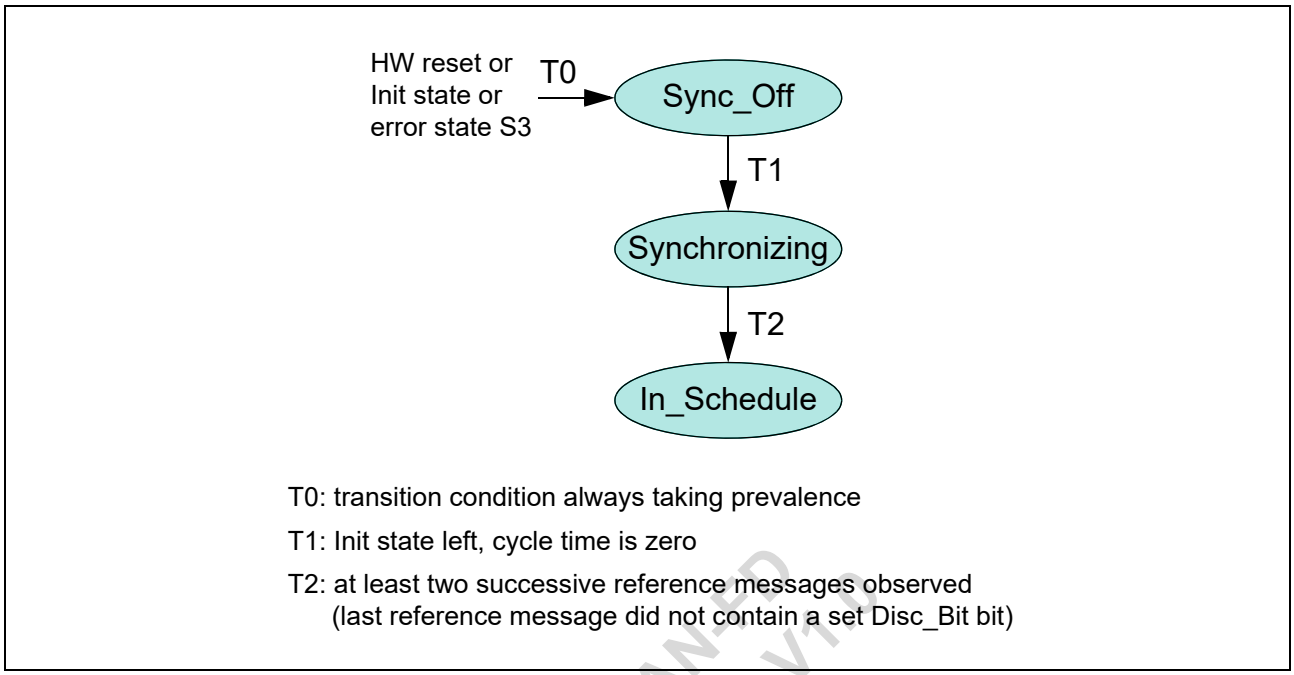



**Figure 37.27 Level 0 schedule synchronization state machine**

Bosch CAN-FD Specification V1.0

### (2) Handling of Error Levels

During Level 0 operation only the following error conditions may occur:

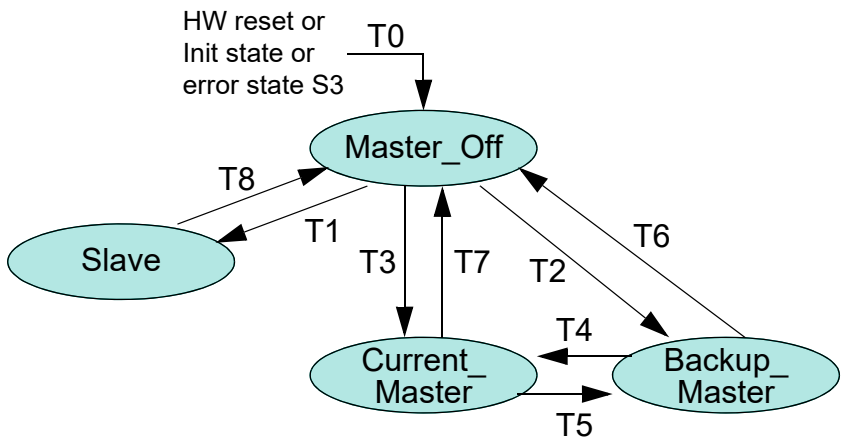
- Watch_Trigger_Reached (S3), reached cycle time $FF00_H$
- CAN_Bus_Off (S3)

Since no S1 and S2 error are possible, the error level can only switch between S0 (No Error) and S3 (Severe Error). In TTCAN Level 0 an S3 error is handled differently. When error level S3 is reached, both MTTCAN0TTOST.SYS and MTTCAN0TTOST.MS are reset, and interrupt flags MTTCAN0TTIR.GTE and MTTCAN0TTIR.GTD are set.

When error level S3 (MTTCAN0TTOST.EL = “11”) is entered, bus monitoring mode is, contrary to TTCAN Level 1 and Level 2, not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

### (3) Master Slave Relation

**Figure 37.28** below describes the master slave relation in TTCAN Level 0. In case of an S3 error the M_TTCAN returns to state Master_Off.




T0: transition condition always taking prevalence

T1: reference message observed when not potential time master

T2: reference message observed with master priority ≠ own master priority, error state ≠ S3

T3: reference message observed with master priority = own master priority, error state ≠ S3

T4: reference message observed with own master priority

T5: reference message observed with master priority higher than own master priority

T6: error state S3

T7: error state S3

T8: error state S3

Bosch CAN-FD Specification V1.0

**Figure 37.28 Level 0 master to slave relation**

### 37.6.4.10 Synchronization to external Time Schedule

This feature can be used to synchronize the phase of the M_TTCAN's schedule to an external schedule (e.g. that of a second TTCAN network or FlexRay network). It is applicable only when the M_TTCAN is current time master (MTTCAN0TTOST.MS = "11").

External synchronization is controlled by event trigger input pin m_ttcan_evt (MTTCAN0EVT) . If bit MTTCAN0TTOCN.ESCN is set, a rising edge at pin m_ttcan_evt (MTTCAN0EVT) the M_TTCAN compares its actual cycle time with the target phase value configured by MTTCAN0TTGTP.CTP.

Before setting MTTCAN0TTOCN.ESCN the Host has to adapt the phases of the two time schedules e.g. by using the TTCAN gap control (see **Section 37.6.4.3, TTCAN Gap Control**). When the Host sets MTTCAN0TTOCN.ESCN, MTTCAN0TTOST.SPL is set.

If the difference between the cycle time and the target phase value MTTCAN0TTGTP.CTP at the rising edge at pin m_ttcan_evt (MTTCAN0EVT) is greater than 9 NTU, the phase lock bit MTTCAN0TTOST.SPL is reset, and interrupt flag MTTCAN0TTIR.CSM is set. MTTCAN0TTOST.SPL is also reset (and MTTCAN0TTIR.CSM is set), when another node becomes time master.

If both MTTCAN0TTOST.SPL and MTTCAN0TTOCN.ESCN are set, and if the difference between the cycle time and the target phase value MTTCAN0TTGTP.CTP at the rising edge at pin m_ttcan_evt (MTTCAN0EVT) is less or equal 9 NTU, the phase lock bit MTTCAN0TTOST.SPL remains set, and the measured difference is used as reference trigger offset value to adjust the phase at the next transmitted reference message.

**NOTE**

The rising edge detection at pin m_ttcan_evt (MTTCAN0EVT) is enabled with the start of each basic cycle. The first rising edge triggers the compare of the actual cycle time with MTTCAN0TTGTP.CTP. All further edges until the beginning of the next basic cycle are ignored.

Bosch CAN-FD Specification V1.0

## 37.7 Usage Notes

### 37.7.1 Port sharing between M_CAN IP and RLIN

By port sharing, ASC@CAN functionality can be achieved by sharing the M_TTCAN/M_CAN with the RLIN3.

All MTTCAN channel 0 pins will be shared with RLIN30.

One pair of MCAN channel 0 pins will be shared with RLIN30.

See the **Section 2, Pin Functions** and the **Section 18, LIN/UART Interface (RLIN3)** for details.

Bosch CAN-FD Specification V1.0

## 37.8 CAUTION

| No. | Outline |
|---|---|
| ID1 | M(TT)CAN - Transmitted bit in control field is falsified when using extreme bit time configurations.<br>(Direction of use) |
| ID2 | M(TT)CAN - Message reception and transmission directly after detection of Protocol Exception Event.<br>(Direction of use) |
| ID3 | M(TT)CAN - Change of CAN operation mode during start of transmission.<br>(Direction of use) |
| ID4 | M(TT)CAN - Problem with frame transmission after recovery from Restricted Operation Mode.<br>(Direction of use) |
| ID5 | M(TT)CAN - Setting / resetting CCCR.INIT during frame reception.<br>(Direction of use) |
| ID6 | M(TT)CAN - Frame transmission in DAR mode.<br>(Direction of use) |
| ID7 | M(TT)CAN - Setting M(TT)CANnCCCR.CCE while a Tx scan is ongoing.<br>(Direction of use) |
| ID8 | M(TT)CAN - Needless activation of interrupt IR.MRAF<br>(Direction of use) |
| ID9 | M(TT)CAN - Return of receiver from Bus Integration state after Protocol Exception Event.<br>(Direction of use) |
| ID10 | M(TT)CAN - Data loss in case storage of a received frame has not completed until end of EOF eld is reached<br>(Specification change notice) |
| ID11 | MTTCAN - Evaluation of event trigger input MTTCAN0EVT<br>(Direction of use) |
| ID12 | MTTCAN - Tx Enable Window of minimum length<br>(Direction of use) |
| ID13 | M(TT)CAN - Retransmission in DAR mode due to lost arbitration at the first two identifier bits.<br>(Direction of use) |
| ID14 | M(TT)CAN - Message RAM / RAM Arbiter not responding in time<br>(Direction of use) |

Bosch CAN-FD Specification V1.0

| | | |
|---|---|---|
| ID1 | | M(TT)CAN - Transmitted bit in control field is falsified when using extreme bit time configurations. |
| | Description | When M(TT)CANnBTP.TSEG2 and M(TT)CANnBTP.BRP are both zero and the M(TT)CAN transmits a frame, the FDF bit in CAN FD format (reserved bit in Classic CAN format) in the control eld may be falsied. The effect is different for frames to be transmitted in Classic CAN format and for frames to be transmitted in CAN FD format.<br>Transmission of Classic CAN Frame:<br>When M(TT)CANnBTP.TSEG2 and M(TT)CANnBTP.BRP are both zero and the M(TT)CAN transmits a Classic CAN frame (M(TT)CANnCCCR.CME = "00") with a 29-bit identier where the MSB (ID28) is '1', the reserved bit following the RTR bit will be transmitted recessive instead of dominant while the rest of the frame is transmitted in Classic CAN format.<br>Transmission of CAN FD Frame:<br>When M(TT)CANnBTP.TSEG2 and M(TT)CANnBTP.BRP are both zero and the M(TT)CAN transmits a CAN FD frame with a 29-bit identifier where the MSB (ID28) is '0' or a CAN FD frame with 11-bit identifier, the FDF bit of the frame is transmitted dominant instead of recessive, the rest of the frame is transmitted in Classic CAN format with a falsified DLC.<br><br>Scope:<br>The erratum is limited to the case when in the bit time configuration for Classic CAN operation and the Arbitration Phase in CAN FD operation M(TT)CANnBTP.TSEG2 and M(TT)CANnBTP.BRP are both zero. This configures the time segment after the sample point to the length of one time quantum and the length of the time quantum to one clock period. This is an unusual configuration.<br>Effects:<br>• Transmission of Classic CAN Frame<br>When a Classic CAN frame is received by a CAN FD enabled receiving node it will interpret the falsified reserved bit as FDF bit. If this bit is recessive instead of dominant, the frame will be interpreted as CAN FD frame. In this case the receiving node will respond with an error frame when it detects that the rest of the frame is not in CAN FD format. A strictly Classic CAN receiving node will interpret the recessive FDF bit as reserved bit, ignore its actual value and will receive this frame correctly without detecting an error.<br>• Transmission of CAN FD Frame<br>When the M(TT)CAN wants to transmit a CAN FD frame, it transmits the FDF bit dominant instead of recessive and the rest of the frame in Classic CAN format with a falsified DLC. |
| | Workaround | Do not use bit timing configurations where M(TT)CANnBTP.TSEG2 and M(TT)CANnBTP.BRP are both zero for CAN FD communication. |

Renesas CAN FD Specification Rev. 0

| | | |
|---|---|---|
| ID2 | | M(TT)CAN - Message reception and transmission directly after detection of Protocol Exception Event. |
| | Description | When a CAN frame is received with bit FDF and the following res bit both recessive, the protocol controller correctly detects a Protocol Exception Event. Reception of the disturbed message is not finished, the message is discarded.<br>If this happened, two cases have to be distinguished:<br>1. Message reception directly after Protocol Exception Event when the next frame is received interrupt flag M(TT)CANnIR.MRAF is set to '1' although the frame has been received correctly<br>2. Message transmission directly after Protocol Exception Event When a frame is transmitted directly after a Protocol Exception Event, that frame is transmitted with faulty frame format. In this case interrupt flag M(TT)CANnIR.MRAF is not set. The frame will cause an error frame.<br>Only the first message after a Protocol Exception Event is affected, all following messages (received or transmitted) have no problem.<br><br>Scope:<br>The erratum is limited to the case when the reserved bit res after the FDF bit in CAN FD frames is received recessive.<br><br>Effect:<br>Reception directly after Protocol Exception Event:<br>Interrupt flag IR.MRAF is set although there was no problem in accessing the Message RAM. The Message is received correctly.<br>Transmission directly after Protocol Exception Event:<br>Transmission of a frame with faulty frame format. |
| | Workaround | None |
| ID3 | | M(TT)CAN - Change of CAN operation mode during start of transmission. |
| | Description | When M(TT)CANnCCCR.CMR is changed during start of transmission, the following may happen:<br>1. Classic CAN -> CAN FD with bit rate switching:<br>When the Tx Event FIFO is used, bits EDL and BRS of the related Tx Event FIFO element do not match with the transmitted frame type. They signal a CAN FD frame with bit rate switching (both set to one) while a Classic CAN frame was transmitted.<br>2. Classic CAN -> CAN FD without bit rate switching:<br>When the Tx Event FIFO is used, bit EDL of the related Tx Event FIFO element does not match with the transmitted frame type. It signals a CAN FD frame while a Classic CAN frame was transmitted.<br>3. CAN FD with bit rate switching -> CAN FD without bit rate switching:<br>When the Tx Event FIFO is used, bit BRS of the related Tx Event FIFO element does not match with the transmitted frame type. It signals a CAN FD frame without bit rate switching while a CAN FD frame with bit rate switching was transmitted.<br>4. CAN FD without bit rate switching -> CAN FD with bit rate switching:<br>When the Tx Event FIFO is used, bit BRS of the related Tx Event FIFO element does not match with the transmitted frame type. It signals a CAN FD frame with bit rate switching while a CAN FD frame without bit rate switching was transmitted.<br>5. CAN FD with/without bit rate switching -> Classic CAN:<br>M(TT)CANnIR.MRAF is set, the M(TT)CAN switches to Restricted Operation Mode, and the transmission is aborted.<br><br>Scope:<br>The erratum is limited to the case when the CAN operation mode is changed during start of transmission.<br><br>Effects:<br>Tx Event FIFO element faulty (cases 1,2,3,4) or interrupt flag M(TT)CANnIR.MRAF set, Restricted Operation Mode entered, and transmission aborted (case 5). |
| | Workaround | Do not change the CAN operation mode by writing to M(TT)CANnCCCR.CMR aslong as there are pending transmission requests (M(TT)CANnTXBRP.TRPnn ='1'). |

Bosch CAN FD Specification V1.0

| | | |
|---|---|---|
| ID4 | | M(TT)CAN - Problem with frame transmission after recovery from Restricted Operation Mode. |
| | Description | After detecting a Message RAM Access Failure during frame transmission, interrupt flag M(TT)CANnIR.MRAF is set and the M(TT)CAN enters Restricted Operation Mode (M(TT)CANnCCCR.ASM = '1').<br>When the Restricted Operation Mode is left by writing M(TT)CANnCCCR.ASM = '0', it may happen, that the first frame transmitted is send out with unexpected identifier and control field. If this is a valid frame, it may happen that it is accepted and acknowledged by a receiver.<br><br>Scope:<br>The erratum is limited to the case when the M(TT)CAN has entered Restricted Operation Mode due to a Message RAM Access Failure, signalled by interrupt flag M(TT)CANnIR.MRAF.<br><br>Effects:<br>With the next transmission after leaving Restricted Operation Mode by resetting M(TT)CANnCCCR.ASM, a frame with unexpected identifier and control field is transmitted which accidentally might be accepted and acknowledged by a receiver. |
| | Workaround | To recover from Restricted Operation Mode proceed as follows:<br>• Initialize M(TT)CAN and clear all pending transmission requests by writing M(TT)CANnCCCR.INIT = '1', M(TT)CANnCCCR.CCE = '1', M(TT)CANnCCCR.ASM = '0<br>• Restart M(TT)CAN by writing M(TT)CANnCCCR.INIT = '0<br>• Configure the CAN operation mode by writing to M(TT)CANnCCCR.CM<br>• Request the transmissions cancelled by step one |
| ID5 | | M(TT)CAN - Setting / resetting M(TT)CANnCCCR.INIT during frame reception. |
| | Description | When M(TT)CANnCCCR.INIT is set while the M(TT)CAN is receiving a frame, the next received frame after resetting M(TT)CANnCCCR.INIT will cause M(TT)CANnIR.MRAF to be set.<br><br>Scope:<br>The erratum is limited to the case when M(TT)CANnCCCR.INIT is set / reset while the M(TT)CAN is receiving a frame.<br><br>Effects:<br>M(TT)CANnIR.MRAF is set when the first frame after resetting M(TT)CANnCCCR.INIT is received although that frame is received correctly. |
| | Workaround | Set M(TT)CANnCCCR.CCE before resetting M(TT)CANnCCCR.INIT. |
| ID6 | | M(TT)CAN - Frame transmission in DAR mode. |
| | Description | 1.When a message is transmitted while M(TT)CANnCCCR.DAR = '1' (automatic retransmission disabled for messages not transmitted successfully), the Event Type of the corresponding Tx Event FIFO element is ET = "01" instead of ET = "10"<br><br>2.When multiple messages are transmitted sequentially using the same Tx Buffer while M(TT)CANnCCCR.DAR = '1', it may happen that a newly requested transmission is not started when it is requested in the time window starting at the successful completion of the previous message and ending at the end of the intermission phase before the bus is idle again.<br><br>This message is then treated as if it had lost arbitration.<br><br>Scope:<br>The erratum is limited to message transmission when DAR mode is configured. Normal CAN / CAN FD operation is not affected<br><br>Effects:<br>1. The Event Type of the associated Tx Event FIFO element is not correct<br>2. When a message was transmitted successfully from a specific Tx Buffer, a following transmission using the same Tx Buffer and requested in the described time window will not be started. |
| | Workaround | Do not use the same Tx Buffer for consecutive DAR transmissions or wait at least for 4 CAN bit times after successful transmission before requesting the next transmission from the same Tx Buffer. |

Bosch CAN FD Specification V1.0

| | | |
|---|---|---|
| ID7 | | M(TT)CAN - Setting M(TT)CANnCCCR.CCE while a Tx scan is ongoing. |
| | Description | When M(TT)CANnCCCR.CCE is set while a Tx scan is in progress, the Tx Handler FSM stops.<br>After M(TT)CANnCCCR.INIT and M(TT)CANnCCCR.CCE are reset, the Tx Handler FSM does not execute transmission requests. |
| | Workaround | 1) Cancel all pending transmission requests by writing 0hFFFF FFFF to register M(TT)CANnTXBCR<br>2) Issue a clock stop request by setting bit M(TT)CANnCCCR.CSR<br>3) Wait until the M(TT)CAN sets CCCR.INIT and M(TT)CANnCCCR.CSA to one<br>4) First reset M(TT)CANnCCCR.CSR<br>5) Then reset M(TT)CANnCCCR.INIT<br>6) Wait until M(TT)CANnCCCR.INIT is read as zero<br>7) Issue a second clock stop request by setting bit M(TT)CANnCCCR.CSR<br>8) Wait until the M(TT)CAN sets M(TT)CANnCCCR.INIT and M(TT)CANnCCCR.CSA to one<br>9) Set M(TT)CANnCCCR.CCE and reset M(TT)CANnCCCR.CSR |
| ID8 | | M(TT)CAN - Needless activation of interrupt IR.MRAF |
| | Description | During frame reception while the M(TT)CAN is in Error Passive state and the Receive Error Counter has the value M(TT)CANnECR.REC = 127, it may happen, that M(TT)CANnIR.MRAF is set although there was no Message RAM access failure.<br><br>In case of MTTCAN:<br>If enabled, the interrupt INTMTTCANI0 or INTMTTCANI1 is generated.<br><br>In case of MCAN:<br>If enabled, the interrupt INTMCANnI0 or INTMCANnI1 is generated. |
| | Workaround | The Message RAM Access Failure interrupt routine needs to check whether M(TT)CANnECR.RP = '1' and M(TT)CANnECR.REC = 127. In this case reset M(TT)CANnIR.MRAF, no further action required. |
| ID9 | | M(TT)CAN - Return of receiver from Bus Integration state after Protocol Exception Event. |
| | Description | In case a started transmission is aborted shortly before the transmission of the FDF bit, a receiver will detect a recessive FDF bit followed by recessive res bit.<br>In this case receiving M(TT)CANs with Protocol Exception Event Handling enabled will detect a protocol exception event and will enter Bus Integration state. These receivers are expected to leave Bus Integration state after 11 consecutive recessive bits.<br>Instead to start counting of 11 recessive bits directly after entering Bus Integration state, the M(TT)CAN needs to see at least one dominant bit before starting to count the sequence of 11 recessive bits. |
| | Workaround | Set and release M(TT)CANnCCCR.INIT. |

Preliminary Specification Rev. 1.0

| | | |
|---|---|---|
| ID10 | | M(TT)CAN - Data loss in case storage of a received frame has not completed until end of EOF eld is reached |
| | Description | Acceptance filtering and storage of a received frame must be finished until the end of EOF field is reached.<br>In case storage of a received message has not completed until end of the received frame is reached, the following faulty behaviour can be observed:<br>• The last write to the Message RAM to complete storage of the received message is omitted, this data is lost. Applies for data frames with DLC > 0, worst case is DLC = 1.<br>• Rx FIFO: FIFO put index M(TT)CANnRXFnS.FnPI is updated although the last FIFO element holds corrupted data.<br>• Rx Buffer: New Data flag NDATn.NDxx is set although the Rx Buffer holds corrupted data.<br>• Interrupt flag IR.MRAF is not set. |
| | Workaround | To ensure that the acceptance filtering and storage of the received message has finished until the end of EOF field, the CLK_HSB must be set accordingly.<br>For classic CAN communication: |

| | | |
|---|---|---|
| Arbitration Phase [Mbps] | 1 | 0.5 |
| Data Phase [Mbps] | 1 | 0.5 |
| min. CLK_HSB frequency [MHz] | 23.7 | 11.9 |

For CAN-FD communication:

| | | | | |
|---|---|---|---|---|
| Arbitration Phase [Mbps] | 1 | 1 | 0.5 | 0.5 |
| Data Phase [Mbps] | 8 | 4 | 8 | 4 |
| min. CLK_HSB frequency [MHz] | 55.9 | 44.0 | 32.4 | 28.0 |

Bosch CAN-FD Specification V1.0

| | | |
|---|---|---|
| ID11 | | MTTCAN - Evaluation of event trigger input MTTCAN0EVT |
| | Description | When the event trigger input MTTCAN0EVT is activated (rising edge) in a narrow time window shortly after the MTTCAN has started the transmission of a reference message, the reference message is transmitted with the Next_is_Gap flag not set, but MTTCANnTTOST.WFE is set, signalling that a reference message with Next_is_Gap = '1' has been received.<br>Due to this, the MTTCAN transmits a reference message with Next_is_Gap flag not set, but assumes that a gap starts at the end of the basic cycle.<br>The backup time master in the system, not having seen a reference message with Next_is_Gap set, sends the next reference message and the M_TTCAN becomes backup time master. It will resume time mastership when it wins arbitration at the transmission of the following reference message. |
| | Workaround | The synchronization of the cycle time phase to an external event has to be repeated. |
| ID12 | | MTTCAN - Tx Enable Window of minimum length |
| | Description | When the Tx Enable Window is congured to its minimum length of one NTU (MTTCAN0TTMLM.TXEW = "0000"), it may happen, that the requested transmission is not started. |
| | Workaround | Congure the Tx Enable Window to a length of more than one NTU (MTTCAN0TTMLM.TXEW > "0000"). |

| | | |
|---|---|---|
| ID13 | | M(TT)CAN - Retransmission in DAR mode due to lost arbitration at the first two identifier bits. |
| | Description | When the M(TT)CAN is configured in DAR mode (M(TT)CANnCCCR.DAR = '1') the Automatic Retransmission for transmitted messages that have been disturbed by an error or have lost arbitration is disabled. When the transmission attempt is not successful, the Tx Buffer's transmission request bit (M(TT)CANnTXBRP.TRPxx) shall be cleared and its cancellation finished bit (M(TT)CANnTXBCF.CFxx) shall be set.<br><br>When the transmitted message looses arbitration at one of the first two identifier bits, it may happen, that instead of the bits of the actually transmitted Tx Buffer, the M(TT)CANnTXBRP.TRPxx and M(TT)CANnTXBCF.CFxx bits of the previously started Tx Buffer (or Tx Buffer 0 if there is no previous transmission attempt) are written (M(TT)CANnTXBRP.TRPxx = '0', M(TT)CANnTXBCF.CFxx = '1').<br><br>If in this case the M(TT)CANnTXBRP.TRPxx bit of the Tx Buffer that lost arbitration at the first two identifier bits has not been cleared, retransmission is attempted.<br>When the M(TT)CAN looses arbitration again at the immediately following retransmission, then actually and previously transmitted Tx Buffer are the same and this Tx Buffer's M(TT)CANnTXBRP.TRPxx bit is cleared and its M(TT)CANnTXBCF.CFxx bit is set.<br><br>Scope:<br>The erratum is limited to the case when the M(TT)CAN looses arbitration at one of the first two transmitted identifier bits while in DAR mode.<br>The problem does not occur when the transmitted message has been disturbed by an error.<br><br>Effects:<br>In this case it may happen, that the M(TT)CANnTXBRP.TRPxx bit is cleared after the second transmision attempt instead of the first.<br>Additionally it may happen that the M(TT)CANnTXBRP.TRPxx bit of the previously started Tx Buffer is cleared, if it has been set again. As in this case the previously started Tx Buffer has lost M(TT)CAN internal arbitration against the active Tx Buffer, its message has a lower identifier priority. It would also have lost arbitration on the CAN bus at the same position. |
| | Workaround | None |
| ID14 | | M(TT)CAN - Message RAM / RAM Arbiter not responding in time |
| | Description | When the M(TT)CAN wants to store a received frame, and the Message RAM / RAM Arbiter does not respond in time (no or delayed m_(tt)can_aeim_ready), this message cannot be stored completely and it is discarded with the reception of the next message. Interrupt flag M(TT)CANnIR.MRAF is set. It may happen that the next received message is stored incomplete. In this case, the respective Rx Buffer or Rx FIFO element holds inconsistent data. |
| | Workaround | Configure the RAM Watcdog to the maximum expected Message RAM access deley. In case the Message RAM / RAM Arbiter does not respond within this time, the Watchdog Interrupt M(TT)CANnIR.WDI is set. In this case discard the frame received after M(TT)CANnIR.MRAF has been activated. |

Bosch CAN-FD Specification V 1.0

## 37.9 Difference among P1M-C, P1H-C and P1H-CE

There are differences in channels, interrupts, and external pins. For details, please refer to **Table 37.2**, **Table 37.8**, and **Table 37.9**.

Bosch CAN-FD Specification V1.0

| REVISION HISTORY | RH850/P1x-C User's Manual: Hardware |
|---|---|

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 4 | How to use this manual, Description changed: Organization |
| | | 4 | How to use this manual, Description changed: How to read this manual |
| | | Section 1 Overview | |
| | | 43 | Figure 1.1 RH850/P1x-C chip block diagram (P1H-CE), Changed |
| | | 44 | 1.4 Features, Description changed, Item: Pin and Port Functions, Features |
| | | 47 | 1.4 Features, Description deleted, Item: High-Speed Universal Synchronous Receiver / Transmitter, Features |
| | | 49 | 1.4 Features, Bullet changed, Item: FlexRay, Features |
| | | 54 | 1.4 Features, Description changed, Item: RAM Modules, Features |
| | | 56 | 1.5 Difference among P1M-C, P1H-C and P1H-CE, Section moved |
| | | Section 2 Pin Functions | |
| | | 57 | Section 2 Pin Functions, List item deleted |
| | | 57 | 2.1 Pin Connection Diagrams, Section deleted: 2.1.1 Clock Supply |
| | | 58 | 2.1.1 P1M-C (QFP-144) (TOP View), Title changed |
| | | 58 | Figure 2.1 Pin Connection Diagram of P1M-C (QFP-144), Title changed and Figure changed |
| | | 59 | Figure 2.2 Pin Connection Diagram of P1M-C (BGA-292), Changed |
| | | 62 | 2.2.1 Pin List and Function assignment, Description changed, CAUTION |
| | | 72 | Table 2.2 Pin Status, Added, Field BIST (TRSTZ = L), *11, *13 |
| | | 72 | Table 2.2 Pin Status, Description changed, Pin Category: Other Func., Pin Name: CVMOUTZ |
| | | 72 | Table 2.2 Pin Status, Description changed, Pin Category: Other Func., Pin Name: ERROROUTZ |
| | | 73 | Table 2.2 Pin Status, Description changed, Pin Category: JTAG, Pin Name: TDI/LPDI |
| | | 73 | Table 2.2 Pin Status, Added, Pin Category: JTAG, Pin Name: TDI/LPDI |
| | | 73 | Table 2.2 Pin Status, Added, Pin Category: JTAG, Pin Name: TDI/LPDI |
| | | 73 | Table 2.2 Pin Status, Added, Pin Category: JTAG, Pin Name: TDO/LPDO |
| | | 73 | Table 2.2 Pin Status, Description changed, Pin Category: JTAG, Pin Name: TDO/LPDO |
| | | 73 | Table 2.2 Pin Status, Description changed, Pin Category: JTAG, Pin Name: TCK/LPDCLKI |
| | | 73 | Table 2.2 Pin Status, Description changed, Pin Category: JTAG, Pin Name: TMS |
| | | 73 | Table 2.2 Pin Status, Description changed, Pin Category: JTAG, Pin Name: TRSTZ/LPDRSTZ |
| | | 73 | Table 2.2 Pin Status, Description changed, Pin Category: JTAG, Pin Name: RDY |
| | | 73 | Table 2.2 Pin Status, Description changed, Pin Category: AUDR |
| | | 74 | Table 2.2 Pin Status: Note 3 and Note 5 corrected |
| | | 74 | Table 2.2 Pin Status, Description changed, Note 4 |
| | | 74 | Table 2.2 Pin Status, Description changed, Note 8 |
| | | 74 | Table 2.2 Pin Status, Description changed, Note 9 |
| | | 94, 95, 96, 97 | Table 2.5 Recommended Connection of Unused Pins, Description corrected: "registor" → "resistor" |
| | | 99 | 2.3.2.1 Port Category, (2) JTAG Port, Description added |
| | | 100 | Table 2.7 Alternative mode selection table, Changed, Title number |
| | | 100 | 2.3.3 Operation Mode, Description changed, CAUTION |
| | | 101 | 2.3.4 Pin data input/output, (1) Output data, Description changed |
| | | 102 | Figure 2.7 Port Control Logic Block Diagram, Changed |
| | | 103 | 2.4.2.1 Control Register, Changed, Heading |
| | | 103 | 2.4.2.1 Control Register, Description changed, Register: JPIBC0, Existent Bit |
| | | 103 | 2.4.2.1 Control Register, Added, Note |
| | | 103 | 2.4.2.1 Control Register, Description Added |
| | | 103 | 2.4.2.1 Control Register, Description changed, Note 1 |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 104 | 2.4.2.1 Control Register, Changed, Heading |
| | | 104 | 2.4.3.1 Control Register, Added, Note |
| | | 105 | 2.4.4.1 Control Register, Changed, Heading |
| | | 105 | 2.4.4.1 Control Register, Added, Note |
| | | 106 | 2.4.5.1 Control Register, Changed, Heading |
| | | 106 | 2.4.5.1 Control Register, Added, Note |
| | | 107 | 2.4.6.1 Control Register, Changed, Heading |
| | | 107 | 2.4.6.1 Control Register, Added, Note |
| | | 108 | 2.4.7.1 Control Register, Changed, Heading |
| | | 108 | 2.4.7.1 Control Register, Added, Note |
| | | 109 | 2.4.8.1 Control Register, Changed, Heading |
| | | 109 | 2.4.8.1 Control Register, Added, Note |
| | | 110 | 2.4.9.1 Control Register, Changed, Heading |
| | | 110 | 2.4.9.1 Control Register, Register: PIPC6, Existent Bit, Changed |
| | | 110 | 2.4.9.1 Control Register, Added, Note |
| | | 111 | 2.4.10.1 Control Register, Changed, Heading |
| | | 111 | 2.4.10.1 Control Register, Added, Note |
| | | 112 | 2.4.10.1 Control Register, Changed, Heading |
| | | 112 | 2.4.11.1 Control Register, Added, Note |
| | | 113 | 2.4.10.1 Control Register, Changed, Heading |
| | | 113 | 2.4.12.1 Control Register, Added, Note |
| | | 114 | Table 2.10, Title changed and Description changed |
| | | 115 | 2.5 Port Register Description, Description changed, CAUTION |
| | | 117 | Table 2.14 PMn/JPM0 register contents, Title corrected |
| | | 141 | Table 2.41 Input pins that incorporate analog filter type A: Title corrected |
| | | 146 | Figure 2.14 MTTCAN0/MCAN0 CRXD connection diagram, Changed, Figure |
| | | 146 | Figure 2.15 RLIN3 RXD connection diagram, Changed, Figure |
| | | 150 | 2.7.3.2 Digital filters, Changed |
| | | 150 | 2.7.3.2 Digital filters, Changed, Form |
| | | 150 | 2.7.3.2 Digital filters, Changed |
| | | 150 | 2.7.3.2 Digital filters, Changed |
| | | 150 | 2.7.3.2 Digital filters, Changed |
| | | 150 | 2.7.3.2 Digital filters, Changed |
| | | 151 | 2.7.3.2 Digital filters, CAUTIONS, Changed, Form |
| | | 151 | 2.7.3.2 Digital filters, CAUTIONS, Changed, Form |
| | | 158 | 2.8 Difference among P1M-C, P1H-C and P1H-CE, Section moved |
| | | Section 3 CPU System | |
| | | 159 | Figure 3.1 Block Configuration Diagram (maximum configuration of P1H-CE), Figure changed |
| | | 161 | Table 3.1 Features of the RH850G3M Core, Description deleted |
| | | 179 | Table 3.25 PID Register Contents, Position change, Bit: 23 to 8, Description: Bit 8 and Bit 9 |
| | | 203 | (1) Detecting PE guard violation, Description changed, “inform” → “informs” |
| | | 217 | Table 3.75 Register Contents of IPGPMTUM4, Description changed, Bit Position: W0, Function |
| | | 217 | Table 3.75 Register Contents of IPGPMTUM4, Description changed, Bit Position: R0, Function |
| | | 219 | Table 3.77 SEGCONT register contents, Description changed, Bit Position: 9, Function |
| | | 219 | Table 3.77 SEGCONT register contents, Description changed, Bit Position: 8, Function |
| | | 219 | Table 3.77 SEGCONT register contents, Description changed, Bit Position: 6, Function |
| | | 220 | Table 3.77 SEGCONT register contents, Description changed and added, Bit Position: 4, Function |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 220 | Table 3.77 SEGCONT register contents, Description changed and added, Bit Position: 0, Function |
| | | 220 | Table 3.77 SEGCONT register contents, Added, Note 1 |
| | | 222 | (c) Error Factor Retention Register (SEGTYPE), Title changed |
| | | 222 | (c) SEG Error Information Register (SEGTYPE), Description changed |
| | | 223 | (d) SEG Error Information (Master) Register (SEGSIDE), Description changed |
| | | 224 | (e) SEGADDR — SEG Error Information (Address) Register, Description changed |
| | | 225 | (3) SEG Function, Description changed |
| | | 242 | 3.4.1 Synchronization of Store Instruction Completion and Subsequent Instruction Generation, Description changed |
| | | 242 | 3.4.1 Synchronization of Store Instruction Completion and Subsequent Instruction Generation, When the updated results in the control registers and memories are reflected in the instruction fetch of a subsequent instruction: (3) SYNCP |
| | | 243 | 3.4.1 Synchronization of Store Instruction Completion and Subsequent Instruction Generation, Description added |
| | | 248 | Figure 3.5 Area That Needs Attention for Prefetch, Note 2 added |
| | | 249 | 3.4.8 System Register Hazards, Section added |
| | | 251 | 3.7 Difference among P1M-C, P1H-C and P1H-CE, Section moved |
| | | Section 4 Address Map | |
| | | 253 | Figure 4.1 Address Map (P1M-C, P1H-C and P1H-CE) (1/2), Changed, Figure |
| | | 254 | Figure 4.1 Address Map (P1M-C, P1H-C and P1H-CE) (2/2), Changed, Figure |
| | | 256 | 4.3 Error notification for an access to unmapped area, Changed, Reference |
| | | 259 | Table 4.1 List of P-bus area access (3/12), Description changed, Start Address: FFC4 C030H, “Unmapped area” → “System reserved”, “—” → “√” |
| | | 267 | Table 4.1 List of P-bus area access (12/12), Description changed, Start Address: FFFF $5000_H$, “FFFF $6FFF_H$” → “FFFF $7AFF_H$”, “Unmapped area” → “System reserved”, “—” → “√” |
| | | 267 | Table 4.1 List of P-bus area access, Description changed, Note 1 |
| | | 267 | Table 4.1 List of P-bus area access, Added, Note 2 |
| | | 271 | 4.5 Difference among P1M-C, P1H-C and P1H-CE, Title changed and Section moved |
| | | 298 | Table 6.14 EIBD Register Contents, Bit Position: 26 to 24, Function: “BCP[TBD]” → “BCP[2:0]” |
| | | 306 | 6.7 Return from Interrupts, Description changed |
| | | 329 | Table 6.26 Difference among P1M-C, P1H-C and P1H-CE (7/7), Changed, Note 1 |
| | | Section 7 DMA | |
| | | 358 | 7.2.6.6 DTS Trigger Select function, Description corrected |
| | | 363 | 7.4.1 Type of Error, Description corrected |
| | | 409 | 7.9.2.23 DTSnnnCM — DTS Channel Master Setting Register (nnn = 000 to 127), Description changed |
| | | 410 | Table 7.45 DMAC Channel Register Address, Added, *2 |
| | | 410 | Table 7.45 DMAC Channel Register Address, Added, *1 |
| | | 410 | Table 7.45 DMAC Channel Register Address, Changed, "Note" → "Note 2" |
| | | Section 8 Reset Controller | |
| | | 472 | Table 8.1 Reset Categories and Reset Sources, Desctiption added, Reset Category: Debug Reset |
| | | 472 | Table 8.1 Reset Categories and Reset Sources, Desctiption changed, Note 1 |
| | | 472 | Table 8.1 Reset Categories and Reset Sources, Added, Note 2 |
| | | 473 | Table 8.2 Reset categories and initialized area ( √ : reset (initialized) — : without reset (kept) ), Description changed, Initialized Area(module): Peripheral |
| | | 475 | Table 8.4 List of Registers, Added, Address: FFF8 1000h, Value after reset: *1 |
| | | 475 | Table 8.4 List of Registers, Added, Address: FFF8 1000h, Value after reset: 0000 0406h or 0000 0404h*2 |
| | | 476 | Table 8.4 List of Registers, added, Note 1 and Note 2 |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 477 | Table 8.5 Register reset condition, Changed, Register Nume: RESF, System Reset 1, “*1” → “*2” |
| | | 477 | Table 8.5 Register reset condition, Note 2 added |
| | | 478 | 8.3.1 RESF — Reset Factor Register, Description added |
| | | 478 | 8.3.1 RESF — Reset Factor Register, Value after reset, Changed |
| | | 478 | 8.3.1 RESF — Reset Factor Register, Added, *2 (Open; *1) |
| | | 478 | 8.3.1 RESF — Reset Factor Register, Added, *1 |
| | | 478 | 8.3.1 RESF — Reset Factor Register, Changed, Bit: 6, Value after reset |
| | | 478 | Table 8.6 Reset Factor Register Contents, Description changed, Bit Position: 10, Function |
| | | 478 | Table 8.6 Reset Factor Register Contents, Description changed, Bit Position: 2, Function |
| | | 479 | Table 8.6 Reset Factor Register Contents (2/2), Bit Position: 1, Bit Name: SRESF0, Added, *2 |
| | | 479 | Table 8.7 Reset Factor Register Contents, Description changed, Bit Position: 1, Function |
| | | 479 | Table 8.7 Reset Factor Register Contents, Description changed, Bit Position: 0, Function |
| | | 482 | 8.3.3 ICUMRESF — ICUMC Reset Factor Register, Description changed |
| | | 509 | 8.3.30 BOOTCTRL — Boot Control Register, Changed, Reference |
| | | 510 | 8.3.31 STAC_DTSRAM — RAM Initialization Mode Control Register for DTS_RAM, Description deleted |
| | | 512 | 8.3.33 STAC_LM0 — RAM Initialization Mode Control Register for PE1, Description changed |
| | | 513 | 8.3.34 STAC_LM1 — RAM Initialization Mode Control Register for PE2, Description changed |
| | | 519 | 8.4.1 Reset Categories, Description changed, "VCC" → "SYSVCC" |
| | | 520 | 8.4.2 Reset Sources, Description changed, "VCC" → "SYSVCC" |
| | | 522 | 8.4.3 Reset Flags, Description deleted |
| | | 522 | 8.4.5 HW BIST, Description corrected |
| | | 523 | 8.4.7 8.4.7 Start Up of Cores, Changed, Reference |
| | | 523 | 8.4.9 Reset Output (RESETOUTZ), Description changed |
| | | Section 10 Core Voltage Monitor (CVM) | |
| | | 530 | 10.1 Overview of RH850/P1x-C CVM, Section added |
| | | 530 | 10.1.1 Units, Section added |
| | | 530 | 10.1.2 Register Addresses, Section changed |
| | | 530 | 10.1.3 Clock Supply, Section changed |
| | | 530 | 10.1.3 Clock Supply, Added, Note |
| | | 530 | 10.1.4 Input/Output Pins, Section changed |
| | | 530 | 10.1.4 Input/Output Pins, Description changed |
| | | 531 | 10.2.1 Functional Overview, Title deleted |
| | | 531 | 10.2 Overview, Description changed |
| | | 531 | 10.2 Overview, Description changed |
| | | 534 | 10.3.2 Reset of registers, Title added |
| | | 534 | 10.3.2 Reset of registers, Description changed |
| | | 542 | 10.4.1 CVM basic function, Description changed |
| | | 542 | 10.5.1 CVM basic function: NOTE added |
| | | 542 | 10.4.1 CVM basic function, (2) CVMOUTZ pin, Description changed, “power on reset” → “Power On Reset” |
| | | 542 | 10.4.1 CVM basic function, (2) CVMOUTZ pin, Changed, Number of the reference |
| | | 544 | (5) Digital noise filter, Description corrected |
| | | 544 | 10.5.1 CVM basic function: NOTE added |
| | | 545 | 10.4.2 CVM function in FBIST and Serial programing mode (Mask of CVMOUTZ pin and CVM reset), Description changed |
| | | 545 | 10.4.3 CVM diagnosis function, (3) CVMMON register, Description changed |
| | | 546 | Figure 10.4 The example of the flowchart of CVM diagnosis application, Changed |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | Section 11 Temperature Sensor | |
| | | 548 | Table 11.3 Clock Supply: Corrected |
| | | 548 | Table 11.3 Clock Supply: Note corrected |
| | | 559 | 11.4.7 OTS0OTDR — Temperature Data Register: Description corrected |
| | | 559 | 11.4.7 OTS0OTDR — Temperature Data Register: Description corrected |
| | | 560 | 11.4.7 OTS0OTDR — Temperature Data Register: NOTE added |
| | | 563 | 11.4.11 OTS0COEFFRn — Coefficient n Register, Added, Note 1 |
| | | 564 | 11.5.1 Functional Description, Title changed |
| | | 564 | Figure 11.3 Example of Temperature Measurement Operation (Continuous Measurement Mode), Figure Changed |
| | | Section 12 Clock Controller | |
| | | 568 | Table 12.3 Clocks and functional modules, Description changed |
| | | 568 | Table 12.3 Clocks and functional modules, Description changed, “SWDT” |
| | | 568 | Table 12.3 Clocks and functional modules, Deleted, Note 1 |
| | | 569 | Figure 12.1 Clock diagram (user’s perspective), Changed |
| | | 573 | Table 12.5 List of registers, “SWD_OPRUN” → “SWDOPRUN” |
| | | 593 | 12.4.1 Clock Selection for Operating Frequency, Description changed |
| | | 593 | 12.4.1 Clock Selection for Operating Frequency, Description added |
| | | 593 | 12.4.1 Clock Selection for Operating Frequency, Description changed |
| | | 593 | 12.4.1 Clock Selection for Operating Frequency, Description changed, (i = 0; 1) → (x = 0; 1) |
| | | 596 | 12.6 Difference among P1M-C, P1H-C and P1H-CE, Title changed and Section moved |
| | | Section 13 Clock Monitor (CLMA) | |
| | | 597 | Section 13 Clock Monitor (CLMA), Description changed |
| | | 597 | 13.1 Features of RH850/P1x-C CLMA, Title changed and Description deleted |
| | | 597 | 13.1.1 Number of Channels, Added, Section |
| | | 597 | 13.1.2 Register Base Addresses, Added, Title |
| | | 597 | 13.1.2 Register Base Addresses, Description added |
| | | 597 | Table 13.2 Register Base Addresses, Moved |
| | | 597 | Table 13.2 Register Base Addresses, Changed, Heading |
| | | 597 | Table 13.2 Register Base Addresses, Deleted, Heading: Component |
| | | 598 | 13.1.3 Clock Supply, Changed, Section |
| | | 598 | 13.1.4 Reset Sources, Title changed and Section Moved |
| | | 599 | 13.1.5 Internal Input/Output Signals, Added, Section |
| | | 600 | 13.2 Overview, Added, Section |
| | | 600 | Figure 13.1 Block Diagram of the Clock Monitor A, Title changed |
| | | 601 | 13.3 Enabling CLMA, Title changed and Section moved |
| | | 601 | 13.4 Functions, Title changed and Section moved |
| | | 601 | 13.4.1 Detection of abnormal clock frequencies, Added, NOTE |
| | | 601 | 13.4.1 Detection of abnormal clock frequencies, Detection Method, Description changed, (2) |
| | | 603 | 13.4.1 Detection of Abnormal Clock Frequencies, Minimum thresholds, Description corrected |
| | | 603 | 13.4.2 Notification of Abnormal Clock Frequency, Added, Section |
| | | 604 | 13.4.3 Self-Test, Title changed |
| | | 604 | 13.4.3 Self-Test, Description partial deleted |
| | | 604 | Figure 13.5 Selt-Test of the clock monitor, Changed |
| | | 604 | Figure 13.5 Selt-Test of the clock monitor, Changed, Figure Title |
| | | 604 | Figure 13.5 Self-diagnosis of the clock monitor, Deleted, Description after the figure |
| | | 605 | 13.5 Registers, Title changed and Section moved |
| | | 605 | 13.5.2 List of Registers, Description changed |
| | | 605 | Table 13.6 List of Registers, Description changed |
| | | 605 | Table 13.6 List of Registers, Description changed |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 605 | Table 13.6 List of Registers, Description changed |
| | | 608 | 13.5.4 CLMAnCTL0 — CLMAn Control Register 0, Description added, Access: |
| | | 608 | 13.5.4 CLMAnCTL0 — CLMAn Control Register 0, Description added, Address: |
| | | 609 | 13.5.5 CLMAnCMPL — CLMAn comparison register L, Description added, Access: |
| | | 609 | 13.5.5 CLMAnCMPL — CLMAn comparison register L, Description added, Address: |
| | | 609 | 13.5.6 CLMAnCMPH — CLMAn comparison register H, Value after reset, Description added, Access: |
| | | 609 | 13.5.6 CLMAnCMPH — CLMAn comparison register H, Value after reset, Description added, Address: |
| | | 609 | 13.5.6 CLMAnCMPH — CLMAn comparison register H, Description changed, Value after reset: |
| | | 609 | 13.5.6 CLMAnCMPH — CLMAn comparison register H, Value after reset, Changed, Bit 11, 10 |
| | | 610 | 13.5.7 CLMAnPCMD — CLMAn Protection command register, Description added, Access: |
| | | 610 | 13.5.7 CLMAnPCMD — CLMAn Protection command register, Description added, Address: |
| | | 610 | 13.5.8 CLMAnPS — CLMAn Protection status register, Description added, Access: |
| | | 610 | 13.5.8 CLMAnPS — CLMAn Protection status register, Description added, Address: |
| | | 611 | 13.5.9 CLMAnEMU0 — CLMAn Emulation Register 0, Description added, Access: |
| | | 611 | 13.5.9 CLMAnEMU0 — CLMAn Emulation Register 0, Description added, Address: |
| | | 612 | 13.5.10 CLMATEST — Clock Monitor Test register, Description added, Access: |
| | | 612 | 13.5.10 CLMATEST — Clock Monitor Test register, Description added, Address: |
| | | 613 | Table 13.15 CLMATEST Register contents (1/2), Bit Name: MONCLKMSK, Function, Description changed |
| | | 613 | Table 13.15 CLMATEST Register contents (1/2), Bit Name: MONCLKMSK, Function, CAUTION Added |
| | | 614 | 13.5.11 CLMATESTS — Clock Monitor Test Status register, Description added, Access: |
| | | 614 | 13.5.11 CLMATESTS — Clock Monitor Test Status register, Description added, Address: |
| | | 615 | 13.5.12 CLMA5CTL1 — CLMA5 control register 1, Description added, Access: |
| | | 615 | 13.5.12 CLMA5CTL1 — CLMA5 control register 1, Description added, Address: |
| | | 615 | 13.5.12 CLMA5CTL1 — CLMA5 control register 1, Description added, Value after reset: |
| | | 615 | 13.5.12 CLMA5CTL1 — CLMA5 control register 1, Description changed, R/W |
| | | 615 | Table 13.17 CLMA5CTL1 Register contents, Description changed |
| | | 617 | 13.7 Difference among P1M-C, P1H-C and P1H-CE, Title changed and Section moved |
| | | Section 14 Power Down Modes | |
| | | 618 | 14.2.1 HALT Mode, Description changed |
| | | 621 | 14.3.2 MSR_LM2 — Module Standby Register for ICUMC, Added, Access and Address |
| | | 622 | 14.3.3 MSR_LM3 — Module Standby Register for MCAN, Added, Access and Address |
| | | 623 | 14.3.4 MSR_LM4 — Module Standby Register for FlexRay, Added, Access and Address |
| | | 624 | 14.3.5 MSR_LM5 — Module Standby Register for GTM, Added, Access and Address |
| | | 625 | 14.3.6 MSR_LM6 — Module Standby Register for Ethernet, Added, Access and Address |
| | | 626 | 14.3.7 MSR_LM7 — Module Standby Register for SENT, Added, Access and Address |
| | | 627 | 14.3.8 MSR_LM8 — Module Standby Register for HS-USRT, Added, Access and Address |
| | | 628 | 14.3.9 MSR_LM10 — Module Standby Register for CSI-H, Added, Access and Address |
| | | 629 | 14.3.10 MSR_LM11 — Module Standby Register for RLIN3, Added, Access and Address |
| | | 630 | 14.3.11 MSR_LM12 — Module Standby Register for AD, Added, Access and Address |
| | | Section 15 External Memory Controller (MEMC) | |
| | | 632 | Table 15.3 Clock Supply, Title changed |
| | | 632 | Table 15.3 Clock Supply, Added line, Unit Name |
| | | 632 | 15.1.4 Hardware Reset, Section deleted |
| | | 632 | Table 15.5 MEMC External Input / Output Signals, Changed, Heading title |
| | | 643 | 15.5 Difference among P1M-C, P1H-C and P1H-CE, Title changed |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | Section 16 Clocked Serial Interface H (CSIH) | |
| | | 650 | 16.2.1 Functional Overview: Description corrected |
| | | 652 | Figure 16.1 CSIH block diagram: Corrected |
| | | 656 | 16.3.3 CSIHnCTL1 — CSIH control register 1: Register name of bits 15 to 8 in the bit chart, corrected |
| | | 656 | Table 16.14 CSIHnCTL1 register contents: Functional description on bit 19, corrected |
| | | 656 | Table 16.14 CSIHnCTL1 register contents: Functional description on bit 18, added |
| | | 657 | Table 16.14 CSIHnCTL1 register contents: Functional description on bit 16, corrected |
| | | 657 | Table 16.14 CSIHnCTL1 register contents: Functional description on bits 15 to 8, corrected |
| | | 657 | Table 16.14 CSIHnCTL1 register contents, Description changed, Bit Position: 6 |
| | | 657 | Table 16.14 CSIHnCTL1 register contents: Functional description on bit 6, corrected |
| | | 659 | Table 16.17 CSIHnCTL2 register contents: Functional description on bits 15 to 3, corrected |
| | | 660 | 16.3.4 CSIHnCTL2 — CSIH control register 2: Description corrected |
| | | 660 | 16.3.4 CSIHnCTL2 — CSIH control register 2: Description corrected |
| | | 678 | Table 16.29 CSIHnCFGx register contents: Functional description on bits 31 and 30, corrected |
| | | 680 | Table 16.29 CSIHnCFGx register contents (3/5): Functional description on bits 17 and 16, corrected |
| | | 681 | Table 16.29 CSIHnCFGx register contents (4/5): Functional description on bits 17 and 16, corrected |
| | | 681 | Table 16.29 CSIHnCFGx register contents (4/5): Functional description of bits 17 and 16, corrected |
| | | 681 | Table 16.29 CSIHnCFGx register contents: Functional description on bit 15, corrected |
| | | 681 | Table 16.29 CSIHnCFGx register contents: Register name corrected |
| | | 682 | Table 16.29 CSIHnCFGx register contents (4/5): Functional description on bits 14 to 12, corrected |
| | | 683 | Table 16.30 CSIHnBRSx register contents: Functional description on bits 11 to 0 |
| | | 686 | 16.3.18 PMMAnTCTLx — PMM TG control register (x = 0 to 7): Description corrected |
| | | 688 | 16.3.20 PMMAnRX — PMMA TG receive buffer: Register name corrected |
| | | 692 | 16.3.24 PMMAnSTC — PMMA TG status clear register: R/W of bits 31 to 16 (Reserved) in the bit chart, corrected |
| | | 693 | Table 16.39 Notes on Setting Registers: Corrected |
| | | 693 | Table 16.39 Notes on Setting Registers: Corrected |
| | | 693 | Table 16.39 Notes on Setting Registers: Corrected |
| | | 693 | Table 16.39 Notes on Setting Registers: Corrected |
| | | 695 | Table 16.39 Notes on Setting Registers: Corrected |
| | | 698 | 16.4.3.1 CSIHTIC in direct access mode: Description corrected |
| | | 709 | 16.5.3.1 Configuration registers: Description corrected |
| | | 710 | Figure 16.14 Chip select and RCB example: Corrected |
| | | 712 | Figure 16.16 to Figure 16.18: Title corrected |
| | | 713 | Figure 16.16 to Figure 16.18: Title corrected |
| | | 716 | Figure 16.22 Baud rate generator block diagram: Corrected |
| | | 716 | 16.5.5 Transmission clock selection: Description corrected |
| | | 717 | 16.5.5 Transmission clock selection: Corrected |
| | | 723 | 16.5.8.2 Data length greater than 16 bits: NOTES corrected |
| | | 726 | 16.5.10.2 CSIHTSSO operation, Changed, Figure |
| | | 733 | 16.5.12.3 Time-out error: Register name corrected |
| | | 740 | Figure 16.45 Enforced Chip Select Idle Example, Changed |
| | | 744 | 16.5.17 Limited Reset and Module stand-by: Description corrected |
| | | 745 | 16.5.18 Chip select Return to Inactive: Corrected |
| | | 745 | 16.5.18 Chip select Return to Inactive: Note1 corrected |
| | | 745 | 16.5.18 Chip select Return to Inactive, Changed, Figure |
| | | 747 | 16.5.19 Select the timing of outputting INT_CSIHTIC: Added |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 757 | Figure 16.55 Master in dual buffer mode, CSIHnCTL1.CSIHnJE = 0, Changed |
| | | 758 | 16.6.3.1 Transmit/receive in master mode when job mode is disabled, Description changed, Procedure |
| | | 759 | Figure 16.56 Master in dual buffer mode, CSIHnCTL1.CSIHnJE = 1, Changed |
| | | 760 | 16.6.3.2 Transmit/receive in master mode when job mode is enabled, Changed, Procedure |
| | | 761 | Figure 16.57 Slave in dual buffer mode, CSIHnCTL1.CSIHnJE = 0, Changed |
| | | 762 | 16.6.3.3 Transmit/receive in slave mode when job mode is disabled, Description changed, Procedure |
| | | 764 | 16.6.4.1 Transmit/receive in master mode when job mode is disabled, Description changed, Procedure: 5 |
| | | 764 | 16.6.4.1 Transmit/receive in master mode when job mode is disabled: Description corrected |
| | | 764 | 16.6.4.1 Transmit/receive in master mode when job mode is disabled, Description deleted, Procedure: 10, 11 |
| | | 777 | Table 16.53 PMM External trigger sources: Title corrected |
| | | 781 | 16.7.4.3 Priority Handler: Description corrected |
| | | 785 | Figure 16.68 Receive Handler flow: Corrected |
| | | 785 | 16.7.4.5 Receive Handler: NOTE corrected |
| | | 786 | 16.7.5.1 TG Preconfiguration: Description corrected |
| | | 788 | Figure 16.70 Power off flow: Corrected |
| | | 788 | 16.7.5.1 TG Preconfiguration: Description corrected |
| | | 788 | 16.8 Difference among D3, D4, D5 and D5ED: Description corrected |
| | | Section 17 High Speed USRT (HS-USRT) | |
| | | 791 | Table 17.6 HS-USRTn Input/Output Signals, Item added |
| | | 825 | 17.4.5.1 Interrupt Sources, Changed, Reference |
| | | 831 | Figure 17.18 Example of Transmission Setting Flow, Changed |
| | | 832 | Figure 17.19 Example of Reception Setting Flow, Changed |
| | | Section 18 LIN/UART Interface (RLIN3) | |
| | | 838 | 18.1.1 Units and Channels, Description moved |
| | | 838 | Table 18.3 Index, Changed, Index: n and b, Meaning |
| | | 840 | Table 18.6 Interrupt Requests, Added |
| | | 841 | Table 18.8 External Input/Output Signals, Changed, Note |
| | | 856 | No.1998_18.3.2.7 RLN3nLSC — LIN Space Configuration Register (n = 0 to 3), IBS[1:0] Bits (Inter-Byte Space Select), Description added |
| | | 870 | 18.3.2.16 RLN3nLDFC — LIN Data Field Configuration Register (n = 0 to 3), FSM Bit (Frame Separate Mode Select), Changed, Reference |
| | | 898 | 18.3.3.18 RLN3nLDBRb — LIN Data Buffer b Register (n = 0 to 3, b = 1 to 8), Title changed |
| | | 916 | 18.3.4.15 RLN3nLDBRb — UART Data Buffer b Register (n = 0 to 3, b = 1 to 8), Title changed |
| | | 926 | Table 18.72 Transition Condition of Each Mode, Description changed |
| | | 940 | 18.5.3.3 Data Transmission/Reception, (1) Data Transmission, Added, Reference |
| | | 960 | Figure 18.26 Transmission Wait Function (If Transmission Data is Set during the Stop Bits in the Received Data), Changed, Note |
| | | 978 | Table 18.94 UART Baud Rate Setting Examples (when LIN communication clock source = 40 MHz), Changed |
| | | 981 | 18.6 Difference among P1M-C, P1H-C and P1H-CE, Added, Section |
| | | Section 19 CAN Controller (MCAN) | |
| | | 982 | Section 19 CAN Controller (MCAN): Description corrected |
| | | 982 | 19.1 Features: Description corrected |
| | | 982 | Table 19.1 M_CAN Specifications: Corrected |
| | | 983 | Table 19.2 Number of MTTCAN/MCAN channel, Deleted, Note 2 |
| | | 984 | Table 19.4 Clock Supply: Note 2 corrected |
| | | 984 | Table 19.4 Clock Supply: Note 2 corrected |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 984 | Table 19.5 Minimune MCAN / MTTCAN clock speed: Corrected |
| | | 986 | 19.2.1 Functional overview: Description corrected |
| | | 987 | 19.2.1 Functional overview: Description corrected |
| | | 988 | Figure 19.1 MCAN/MTTCAN block diagram, Changed |
| | | 988 | 19.2.2 Block Diagram: Description deleted |
| | | 988 | 19.2.2 Block Diagram: Note 1 deleted |
| | | 990 | 19.4.1 Procedure of Module Standby and Limited Reset: Description corrected |
| | | 992 | 19.5.1 Overview: Description corrected |
| | | 992 | 19.5.1.1 Features: Description corrected |
| | | 995 | 19.6.2.2 Register Map: Description corrected |
| | | 995 | Table 19.12 M_CAN Register Map: Corrected |
| | | 1005 | Table 19.20 MCANnNBTP Register Contents: Bit position and functional description on bits 6 to 0, corrected |
| | | 1049 | Figure 19.4 Message RAM Configuration: Corrected |
| | | 1053 | Table 19.61 Tx Buffer Element: Corrected |
| | | 1064 | 19.5.3.1 Operating Modes, (4) Transmitter Delay Compensation: Titile corrected |
| | | 1064 | (4) Transceiver Delay Compensation, (a) Description: Description corrected |
| | | 1065 | Figure 19.5 Transmitter delay measurement: Title corrected |
| | | 1078 | 19.6.3.4 Rx Handling: Description corrected |
| | | 1087 | 19.6.1 Overview: Description corrected |
| | | 1088 | 19.6.1.1 Features: Description corrected |
| | | 1089 | Figure 19.15 M_TTCAN Block Diagram: Title corrected |
| | | 1091 | 19.6.1.3 Dual Clock Sources: Description corrected |
| | | 1092 | 19.6.2.1 Hardware Reset Description: Description corrected |
| | | 1094 | 19.6.2.2 Register Map, (1) Access to reserved Register Addresses: Description corrected |
| | | 1101 | Table 19.78 MCANnCCCR Register Contents: Functional description on bit 3, corrected |
| | | 1102 | Table 19.79 MTTCAN0NBTP Register Contents: Functional description on bits 6 to 0, corrected |
| | | 1109 | Table 19.85 MTTCAN0PSR Register Contents: Functional description on bits 2 to 0, corrrected |
| | | 1110 | Table 19.86 MTTCAN0TDCR Register Contents: Functional description on bits 14 to 8, 6 to 0, corrected |
| | | 1168 | Figure 19.16 Message RAM Configuration: Corrected |
| | | 1173 | 19.6.2.4 Message RAM, (3) Tx Buffer Element: NOTE corrected |
| | | 1178 | 19.6.2.4 Message RAM, (5) Standard Message ID Filter Element: Description corrected |
| | | 1180 | 19.6.2.4 Message RAM, (6) Extended Message ID Filter Element: Description corrected |
| | | 1184 | 19.6.3.1 Operating Modes, (2) Normal Operation: Description corrected |
| | | 1185 | 19.6.3.1 Operating Modes, (4) Transceiver Delay Compensation: Description corrected |
| | | 1186 | 19.6.3.1 Operating Modes, (4) Transceiver Delay Compensation, (a) Description: Description corrected |
| | | 1187 | Figure 19.17 Transmitter delay measurement: Title corrected |
| | | 1189 | 19.6.3.1 Operating Modes, (6) Bus Monitoring Mode: Description corrected |
| | | 1189 | 19.6.3.1 Operating Modes, (7) Disabled Automatic Retransmission: Description corrected |
| | | 1190 | 19.6.3.1 Operating Modes, ((8) Power Down (Sleep Mode): Description corrected |
| | | 1190 | 19.6.3.1 Operating Modes, (8) Power Down (Sleep Mode): Description corrected |
| | | 1191 | 19.6.3.1 Operating Modes, (9) Test Modes, (b) Internal Loop Back Mode: Description corrected |
| | | 1200 | 19.6.3.4 Rx Handling, (4) Debug on CAN Support, Description changed |
| | | 1234 | 19.7 Usage Notes, Section moved |
| | | 1234 | 19.8 CAUTION, Section moved |
| | | 1234 | 19.10 CAUTION: Description corrected |
| | | 1235 | 19.10 CAUTION: Description corrected |
| | | 1236 | 19.9 Difference among P1M-C, P1H-C and P1H-CE, Section moved |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | Section 20 FlexRay (FLX) | |
| | | 1237 | 20_FlexRay, Added, NOTE |
| | | 1246 | Table 20.7 FlexRay Register Map (3/5), Changed, Address, FlexRay Even Sync ID Register m |
| | | 1251 | 20.2.2.1 FLXAnFROC — FlexRay Operation Control Register, Description changed, (5) FLXAnFROC.BEC |
| | | 1257 | Table 20.11 FLXAnFREIR Register Contents (2/2), Description changed, Function, CCL |
| | | 1257 | Table 20.11 FLXAnFREIR Register Contents (2/2), Description changed, Function, CCL |
| | | 1257 | 20.2.4.1 FLXAnFREIR — FlexRay Error Interrupt Register, Description added, (1) FLXAnFREIR.TABB |
| | | 1264 | 20.2.4.2 FLXAnFRSIR — FlexRay Status Interrupt Register, Changed, (6) FLXAnFRSIR.MBSI |
| | | 1266 | 20.2.4.2 FLXAnFRSIR — FlexRay Status Interrupt Register, Changed, (16) FLXAnFRSIR.RXI |
| | | 1266 | 20.2.4.2 FLXAnFRSIR — FlexRay Status Interrupt Register, Changed, (17) FLXAnFRSIR.TXI |
| | | 1268 | Table 20.13 FLXAnFREILS Register Contents (2/2), Space added, SFBML |
| | | 1289 | FLXAnFRSUCC1 — FlexRay SUC Configuration Register 1, Changed, Bit 7 |
| | | 1357 | 20.2.9.1 FLXAnFRMHDS — FlexRay Message Handler Status Register, Changed |
| | | 1357 | 20.2.9.1 FLXAnFRMHDS — FlexRay Message Handler Status Register, Changed |
| | | 1359 | 20.2.9.1 FLXAnFRMHDS — FlexRay Message Handler Status Register, Changed |
| | | 1359 | 20.2.9.1 FLXAnFRMHDS — FlexRay Message Handler Status Register, Changed |
| | | 1371 | 20.2.11 Input Buffer, Changed |
| | | 1419 | 20.2.14.2 FLXAnFROTS — FlexRay Output Transfer Status Register, Description deleted, (4) FLXAnFROTS.OWP |
| | | 1424 | 20.2.14.3 FLXAnFRAES — FlexRay Access Error Status Register, Description changed, (5) FLXAnFRAES.EIDX |
| | | 1429 | Table 20.95 NTU0CFG Register Contents, Changed, Bit Position |
| | | 1447 | 20.3.2.5 Configuration of NIT Start and Offset Correction Start, Changed |
| | | 1447 | 20.3.2.5 Configuration of NIT Start and Offset Correction Start, Changed |
| | | 1447 | 20.3.2.5 Configuration of NIT Start and Offset Correction Start, Changed |
| | | 1449 | 20.3.4.1 Global Time, Changed |
| | | 1449 | 20.3.4.2 Local Time, Format changed |
| | | 1449 | 20.3.4.2 Local Time, Changed |
| | | 1468 | 20.3.8.1 Slot Counter Filtering, Changed |
| | | 1490 | 20.3.13.2 Data Partition, Table number changed |
| | | 1490 | 20.3.13.2 Data Partition, Table number changed |
| | | 1504 | 20.3.16.1 Input Data Transfer, Changed, (3) Input pointer table |
| | | 1516 | 20.3.16.2 Output Data Transfer, Changed, (7) Transfer function of user output transfer requests |
| | | 1520 | 20.3.16.5 Behaviors in Case of RAM Read Errors, Changed, (1) Read error during transfer from TBF to MBF |
| | | 1520 | 20.3.16.5 Behaviors in Case of RAM Read Errors, Changed, (1) Read error during transfer from TBF to MBF |
| | | 1522 | 20.3.17.1 Little Endian Alignment, Changed, (2) FLXAnFRWRDSx (x = 1 to 64) |
| | | 1522 | 20.3.17.1 Little Endian Alignment, Changed, (3) |
| | | 1523 | 20.3.17.1 Little Endian Alignment, Description changed, (3) |
| | | 1523 | 20.3.17.1 Little Endian Alignment, Changed, (3) |
| | | 1523 | 20.3.17.1 Little Endian Alignment, Changed, (3) |
| | | 1523 | 20.3.17.1 Little Endian Alignment, Changed, (3) |
| | | 1523 | 20.3.17.1 Little Endian Alignment, Changed, (3) |
| | | 1523 | 20.3.17.1 Little Endian Alignment, Description changed, (3) |
| | | 1523 | 20.3.17.2 Big Endian Alignment, Changed, (3) |
| | | 1523 | 20.3.17.2 Big Endian Alignment, Changed, (3) |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 1523 | 20.3.17.2 Big Endian Alignment, Changed, (3) |
| | | 1523 | 20.3.17.2 Big Endian Alignment, Changed, (3) |
| | | 1523 | 20.3.17.2 Big Endian Alignment, Changed, (3) |
| | | 1523 | 20.3.17.2 Big Endian Alignment, Changed, (3) |
| | | 1523 | 20.3.17.2 Big Endian Alignment, Changed, (3) |
| | | 1526 | Figure 20.35 The NTU loss monitor operation in compare unit (in case of loss 1), Description changed |
| | | 1526 | Figure 20.36 The NTU loss monitor operation in compare unit (in case of loss 2), Description changed |
| | | 1528 | 20.4 CAUTION, Section moved |
| | | 1530 | 20.6 Difference among P1M-C, P1H-C and P1H-CE, Section moved |
| | | Section 21 Ethernet Controller (ETNA) | |
| | | 1533 | Table 21.6 Ethernet External Input / Output Signals: Corrected |
| | | 1554 | 21.3.9 ETNAnPIRm — PHY Interface Register m, Value after reset: “0000 000$X_H$” → “0000 0000$_H$” |
| | | 1569 | Table 21.32 ETNAnAPRm register contents, Description changed |
| | | 1582 | Table 21.45 ETNAnTSU_FWINMK register contents, Description changed, Bit: 20 to 16, “MACm” → “MAC0” |
| | | 1582 | Table 21.45 ETNAnTSU_FWINMK register contents, Description changed, Bit: 4 to 0, “MACm” → “MAC1” |
| | | 1628 | Table 21.79 ETNAnRFSRm register contents, Description changed |
| | | 1628 | Table 21.79 ETNAnRFSRm register contents, Description changed |
| | | 1632 | 21.4.6 Transmission Clock, Description deleted |
| | | 1656 | Figure 21.14 MAC Receiver State Transitions, Figure replace, High-definition |
| | | 1662 | Figure 21.16 E-DMAC Operation of multi-buffer frame transmission in store and forward mode, Figure replace, High-definition |
| | | 1665 | 21.4.36 Interrupt Sources, Description changed |
| | | 1666 | Table 21.88 List of ETNA Interrupts and detected statuses (2/3), Changed, Interrupt/Status, Transmit/receive interrupt for port 1 (INTETNA0) |
| | | 1667 | Table 21.88 List of ETNA Interrupts and detected statuses (2/3), Changed, Interrupt/Status, Transmit/receive interrupt for port 1 (INTETNA0) |
| | | Section 22 Single Edge Nibble Transmission (SENT) | |
| | | 1683 | Figure 22.1 Block Diagram of RSENT, Changed |
| | | 1691 | Table 22.12 BRP register contents, Title changed |
| | | 1701 | 22.3.9 RSENTnCS — RSENT Communication Status Register, RSENTnCS.SMS (Slow Channel Message Lost Status), Description changed |
| | | 1727 | 22.4.1.5 Register Behavior in Operation Modes, Changed, Reference |
| | | 1728 | (2) Timestamp Counter Operation, Description changed |
| | | 1731 | 22.4.2.2 Communication Clock Configuration, (1) RX BRP Setting, Description changed |
| | | 1731 | (2) RX and SPC Tick Settings, Corrected |
| | | 1735 | 22.4.3.2 Message Reception, (2) Data Nibble Reception, Changed, Reference |
| | | 1737 | (3) Fast Channel Message Reception, Description deleted |
| | | 1746 | 22.4.3.2 Message Reception, (8) Error Flagging, Changed, Reference |
| | | 1751 | 22.4.5 Interrupts and Checks, Changed, Reference |
| | | 1752 | 22.5 Difference among P1M-C, P1H-C and P1H-CE, Section moved |
| | | Section 23 Window Watchdog Timer A (WDTA) | |
| | | 1756 | Figure 23.1 Block diagram of the Window Watchdog Timer A, Corrected |
| | | 1757 | Table 23.6 Register list, Description changed, Register name |
| | | 1757 | Table 23.6 Register list, Description changed, Register name |
| | | 1757 | Table 23.6 Register list, Changed, Note |
| | | 1758 | 23.3.2 WDTAnWDTE — WDTA enable register, Changed, Reference |
| | | 1758 | 23.3.2 WDTAnWDTE — WDTA enable register, Description changed |
| | | 1758 | Table 23.11 WDTAnWDTE behavior, Description changed |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 1758 | Table 23.11 WDTAnWDTE behavior, Description changed |
| | | 1758 | Table 23.11 WDTAnWDTE behavior, Description changed |
| | | 1761 | Table 23.16 WDTAnREF register contents, Title changed |
| | | 1762 | Table 23.17 WDTA start-up options, Description corrected |
| | | 1764 | 23.4.2.4 Default start mode timing, (2), Description changed |
| | | 1767 | Figure 23.4 Timing diagram of WDTA error signal to ECM module, Corrected |
| | | 1769 | Figure 23.6 Timing diagram of WDTA window function, Changed |
| | | 1770 | 23.6 Difference among P1M-C, P1H-C and P1H-CE, Section moved |
| | | Section 24 System Timer (STM) | |
| | | 1774 | Figure 24.1 STMn block diagram, Changed |
| | | 1782 | Table 24.14 STMnIS register contents, Description changed |
| | | 1782 | Table 24.14 STMnIS register contents, Added, Bit Position: 15 to 0 |
| | | 1790 | Table 24.23 Configurable Interrupt vector assignment, Corrected, Interrupt source |
| | | 1795 | Figure 24.7 Compare match event conflicting with counter stop, Title change |
| | | 1796 | 24.4.3.4 Confliction of flag set with emulation mode start, Corrected |
| | | 1798 | Figure 24.10 Access to register, Title changed |
| | | 1799 | 24.4.5 Resets, Corrected |
| | | 1800 | 24.5 Difference among P1M-C, P1H-C and P1H-CE, Section moved |
| | | Section 25 Generic Timer Module (GTM) | |
| | | 1801 | Table 25.1 IP version, Changed, Heading: P1M-C (Config 2) |
| | | 1801 | Table 25.1 IP version, Changed, Heading:P1H-C, P1H-CE (Config 3) |
| | | 2235 | 25.21 CAUTION, Section moved |
| | | 2238 | 25.22 Difference among P1M-C, P1H-C and P1H-CE, Section moved |
| | | Section 26 Peripheral Interconnect (PIC) | |
| | | 2241 | Figure 26.1 Block diagram of PIC, Added |
| | | 2258 | Figure 26.4 Block diagram for GTM output monitoring function, Changed |
| | | Section 27 A/D Converter (ADCF) | |
| | | 2261 | Table 27.4 Clock Supply: Note added |
| | | 2266 | Figure 27.1 Block Diagram (CH0), Description changed, Note 2 |
| | | 2267 | Figure 27.2 Block Diagram (CH1), Description changed, Note 2 |
| | | 2268 | 27.2.3 Physical Channels, Virtual Channels and Scan Groups, Description changed |
| | | 2274 | 27.3.4 ADCFnVCRj — Virtual Channel Register j, Description changed, CAUTION: [NOTE] |
| | | 2276 | 27.3.5 ADCFnDRj — Data Register j, Changed, Figure |
| | | 2277 | 27.3.6 ADCFnDIRj — Data Supplementary Information Register j, Changed, Address |
| | | 2277 | 27.3.6 ADCFnDIRj —Data Supplementary Information Register j, Changed, Bit Name |
| | | 2277 | Table 27.13 ADCFnDIRj register contents, Changed, Bit Name |
| | | 2281 | Table 27.17 ADCFnMPXCURR register contents, Description changed, Bit Position: 3 to 0, Function |
| | | 2282 | Table 27.18 ADCFnMPXOWR register contents, Description changed |
| | | 2284 | 27.3.13 ADCFnADENDP0 — A/D Conversion Monitor Virtual Channel Pointer: Description corrected |
| | | 2284 | Table 27.20 ADCFnADENDP0 register contents: Functional description on bits 5 to 0, corrected |
| | | 2302 | 27.3.29 ADCFnSGSRx — Scan Group x Status Register, Added. CAUTION |
| | | 2314 | Figure 27.11 diagnosis circuits, Added, Note |
| | | 2314 | Figure 27.11 diagnosis circuits, Changed |
| | | 2315 | 27.4.7.2 A/D Conversion Circuit Diagnostic Function, Change the form, CAUTION |
| | | 2317 | 27.4.7.3 Wiring-Break Detection Diagnostic Function, Description changed, NOTE |
| | | 2319 | 27.4.11 Starting a Scan Group by Using a Hardware Trigger, Description Added, NOTE |
| | | 2319 | Figure 27.15 A/D Conversion Monitor Timing, Changed |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | Section 28 Functional Safety | |
| | | 2333 | Table 28.3 Code Flash ECC: Corrected |
| | | 2334 | Table 28.4 List of Register: Corrected |
| | | 2335 | (1) UCFERRINT — FLI (Code-Flash) Data and Address Error Information Control Register: Title and description corrected |
| | | 2341 | (7) UCFnSEDADR — FLI (Code-Flash) n ECC SED Address Register (n = 1 to 4): Description corrected |
| | | 2346 | (4) Self-diagnosis of address parity check function: NOTE added |
| | | 2354 | (6) DFERRINTn — Data flash error notification control register (n = 0, 1): Description corrected |
| | | 2355 | (7) DFEADRn — Data flash 1st error address register (n = 0, 1): Description corrected |
| | | 2355 | Table 28.24 DFEADRn register contents: Functional description on bits 20 and 19, corrected |
| | | 2355 | Table 28.24 DFEADRn register contents: Functional description on bits 20 and 19, corrected |
| | | 2364 | (2) LRERRINT_PE1/PE2 — Local-RAM Error Information Control Register: Description corrected |
| | | 2377 | 28.2.5.1 Overview, Description corrected |
| | | 2377 | 28.2.5.1 Overview, Description changed |
| | | 2377 | 28.2.5.1 Overview, Description deleted |
| | | 2378 | Table 28.40 Global RAM ECC, Description changed: "EDC" → "ECC" |
| | | 2378 | Table 28.40 Global RAM ECC, Changed, Item: Address ECC |
| | | 2381 | Table 28.41 List of Registers, Added, Note 1 |
| | | 2386 | Table 28.46 UGRSERSTR register contents, Description corrected |
| | | 2391 | (9) GRECCCTL_GRAMC — Global-RAM ECC Control Register, Description changed |
| | | 2394 | (11) GRTDATBF[0/1][L/H] — Global-RAM Test Data Read Buffer (Lower/Upper 32 bit of Bank0/1): NOTE added |
| | | 2397 | (14) GRECCCTL_VCI2GRAM/PE1/PE2/AXI2GRAM — Global-RAM ECC Control Register, Description deleted and Note 1 added |
| | | 2417 | (14) ITOVFSTR_PE1/PE2 — Instruction Cache Tag RAM Error Count Overflow Status Register, Description corrected |
| | | 2422 | 28.2.8.1 Overview, (1) FlexRay, CSIHn (n = 0 to 3), MCAN, GTM: Titile corrected |
| | | 2423 | 28.2.8.1 Overview: (3) FlexRay added |
| | | 2426 | Table 28.77 List of Register: Note 3 added |
| | | 2426 | Table 28.78 Register Map: Corrected |
| | | 2430 | (2) TMC — ECC test mode control register: Name of bit 7 in the bit chart, corrected |
| | | 2440 | Table 28.89 List of Registers, Description corrected |
| | | 2440 | Table 28.89 List of Registers, Description corrected |
| | | 2441 | Table 28.89 List of Registers: Register name corrected |
| | | 2441 | Table 28.89 List of Registers: Register name corrected |
| | | 2441 | Table 28.89 List of Registers: Register name corrected |
| | | 2443 | Table 28.89 List of Registers, Description changed: Register Name |
| | | 2443 | Table 28.89 List of Registers, Description changed: Register Name |
| | | 2443 | Table 28.89 List of Registers, Description changed: Register Name |
| | | 2443 | Table 28.89 List of Registers, Description changed: Register Name |
| | | 2444 | Table 28.89 List of Registers: Register name corrected |
| | | 2444 | Table 28.89 List of Registers: Register name corrected |
| | | 2444 | Table 28.89 List of Registers: Register name corrected |
| | | 2445 | Table 28.89 List of Registers: Register name corrected |
| | | 2445 | Table 28.89 List of Registers: Register name corrected |
| | | 2445 | Table 28.89 List of Registers: Register name corrected |
| | | 2445 | Table 28.89 List of Registers: Register name corrected |
| | | 2446 | Table 28.89 List of Registers: Register name corrected |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 2446 | Table 28.89 List of Registers: Register name corrected |
| | | 2449 | Table 28.92 CFERSTCLR_VCI2CFBB register contents, Description changed, Bit Position: 0, Function |
| | | 2451 | (5) CFERSTR_VCI2CFBB — FLI (Code-Flash) Address ECC SED/DED Status Register: Description corrected |
| | | 2455 | Table 28.98 IFERSTCLR_PE1/PE2 register contents, Description changed, Bit Position: 0, Function |
| | | 2461 | Table 28.104 LSSERSTCLR_PE1/PE2 register contents, , Description changed, Bit Position: 0, Function |
| | | 2467 | Table 28.110 LSMERSTCLR_PE1/PE2 register contents, Description changed, Bit Position: 0, Function |
| | | 2472 | (26) VPERRINT_SGn — System Interconnect (PBus I/F) Data Error Information Control Register (n = 0 to 3, 5): Description corrected |
| | | 2473 | Table 28.116 VPERSTCLR_SGn register contents, Description changed, Bit Position: 0, Function |
| | | 2477 | (31) VPECCCTL_TERM_SG7 — PBus Data and Address ECC Control Register: Title corrected |
| | | 2477 | (31) VPECCCTL_TERM_SG7 — PBus Data and Address ECC Control Register: Description corrected |
| | | 2478 | (32) VPERRINT_TERM_SG7 — System Interconnect (PBus I/F) Data and Address Error Information Control Register: Title corrected |
| | | 2479 | Table 28.122 VPERSTCLR_TERM_SG7 register contents, Description changed, Bit Position: 0, Function |
| | | 2481 | (35) VPERSTR_TERM_SG7 — System Interconnect (PBus I/F) Data and Address ECC SED/DED Status Register: Title corrected |
| | | 2485 | Table 28.128 VPERSTCLR_VCI2VPI register contents, Description changed, Bit Position: 0, Function |
| | | 2490 | (44) VCERRINT_PDMA — System Interconnect Data Error Information Control Register: Title and description corrected |
| | | 2491 | Table 28.134 VCERSTCLR_PDMA register contents, Description changed, Bit Position: 0, Function |
| | | 2496 | (50) APERRINT_PFSS — P-Bus Data and Address Error Information Control Register: Titile corrected |
| | | 2497 | Table 28.140 APERSTCLR_PFSS register contents, Description changed, Bit Position: 0, Function |
| | | 2500 | (54) APEADR_PFSS — P-Bus Data and Address ECC SED/DED Address Register, Description corrected |
| | | 2501 | (55) APECnECCCTL — P-Bus Data and Address ECC Control Register: Description corrected |
| | | 2503 | (56) APECnERRINT — P-Bus Data and Address Error Information Control Register: Title corrected |
| | | 2503 | (56) PECnERRINT — P-Bus Data and Address Error Information Control Register: Description corrected |
| | | 2504 | (57) APECnSTCLR — P-Bus Data and Address ECC SED/DED Status Clear Register: Description corrected |
| | | 2504 | Table 28.147 APECnSTCLR register contents, Description changed, Bit Position: 0, Function |
| | | 2505 | (58) APECnOVFSTR — P-Bus Data and Address Error Count Overflow Status Register: Description corrected |
| | | 2506 | (59) APECn1STERSTR — P-Bus Data and Address ECC SED/DED Status Register: Description corrected |
| | | 2507 | (60) APECn1STEADR0 — P-Bus Data and Address ECC SED/DED Address Register: Description corrected |
| | | 2508 | (2) Self-diagnosis of the Data ECC check function (Master), Description changed |
| | | 2510 | 28.2.10.1 ECC code for RAM Modules except Instruction cache RAM (data), (1) Generating an Error Correcting Code, Description corrected |
| | | 2512 | 28.2.10.2 EDC Function for Instruction cache RAM (data), (2) Error Detection, Description corrected |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 2515 | 28.2.10.3 ECC Function for Code Flash, (2) Error Detection, Description corrected |
| | | 2522 | 28.3.2.3 PDMA_COMP_CNTRL — PDMA Comparator Error Injection Control Register: Description corrected |
| | | 2527 | 28.4.1.1 Protection mechanisms inside the processors, (2) Internal Peripheral Guard (IPG), Description changed |
| | | 2544 | Table 28.169 PBG Group: Corrected |
| | | 2547 | Table 28.169 PBG Group: Corrected |
| | | 2550 | Table 28.170 List of Registers, Corrected: R |
| | | 2550 | Table 28.170 List of Registers, Corrected: R |
| | | 2550 | Table 28.170 List of Registers, Corrected: R |
| | | 2550 | Table 28.170 List of Registers, Corrected: R |
| | | 2582 | 28.5 BIST: Description corrected |
| | | 2590 | Table 28.201 MBISTSIG2 register contents: Functional description on bits 19 to 0, corrected |
| | | Section 29 Error Control Module (ECM) | |
| | | 2606 | Table 29.7 Specification Overview, Corrected, Item: Hi-Z control signal trigger, Description |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2609 | Table 29.8 List of Error Inputs (2/10), Changed, No.: 4, Error source explanation |
| | | 2612 | Table 29.8 List of Error Inputs, Corrected |
| | | 2613 | Table 29.8 List of Error Inputs (6/10), Corrected, No.: 29, Error source explanation |
| | | 2613 | Table 29.8 List of Error Inputs, Corrected |
| | | 2617 | Table 29.8 List of Error Inputs, Changed, Error source explanation: No.88 to 91 |
| | | 2617 | Table 29.8 List of Error Inputs, Added, Note 3 and Note 4 |
| | | 2619 | 29.3.5 Writing to Protected Registers, Description added |
| | | 2619 | 29.3.5.1 Protection Unlock Sequence; ALL, Changed, “ECMnPCMD0” → “ECMmnPCMD0” |
| | | 2622 | Table 29.10 Address List of ECM Master and Checker Registers, Changed, Register Symbol: ECMMnESET, Value after reset |
| | | 2622 | Table 29.10 Address List of ECM Master and Checker Registers, Changed, Register Symbol: ECMMnECLR, Value after reset |
| | | 2622 | Table 29.10 Address List of ECM Master and Checker Registers, Changed, Register Symbol: ECMCnESET, Value after reset |
| | | 2622 | Table 29.10 Address List of ECM Master and Checker Registers, Changed, Register Symbol: ECMCnECLR, Value after reset |
| | | 2623 | Table 29.11 Address List of ECM common Registers, Register Symbole: ECMnEPCFG, Access Width |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 2623 | Table 29.11 Address List of ECM common Registers, Register Symbole: ECMnEPCFG, Value after reset |
| | | 2624 | Table 29.11 Address List of ECM common Registers, Register Symbole: ECMnDTMCTL, Access Width |
| | | 2624 | Table 29.11 Address List of ECM common Registers, Register Symbole: ECMnDTMCTL, Value after reset |
| | | 2627 | 29.4.2 ECMmnESET — ECM Master/Checker n Error Set Trigger Register (m = M/C), Bit: 0, Value after reset |
| | | 2628 | 29.4.3 ECMmnECLR — ECM Master/Checker n Error Clear Trigger Register (m = M/C), Bit: 0, Value after reset |
| | | 2628 | 29.4.3 ECMmnECLR — ECM Master/Checker n Error Clear Trigger Register (m = M/C), Description added, CAUTIONS |
| | | 2634 | 29.4.8 ECMnEPCFG — ECM n Error Pulse Configuration Register, Changed, Access |
| | | 2634 | 29.4.8 ECMnEPCFG — ECM n Error Pulse Configuration Register, Changed, Value after reset |
| | | 2634 | 29.4.8 ECMnEPCFG — ECM n Error Pulse Configuration Register, Changed, 32bit register |
| | | 2634 | Table 29.20 ECMnEPCFG register contents, Changed, Bit Position |
| | | 2644 | 29.4.18 ECMnEMK0 — ECM n Error Mask Register 0, Description added, NOTE |
| | | 2645 | 29.4.19 ECMnEMK1 — ECM n Error Mask Register 1, Description added, NOTE |
| | | 2646 | 29.4.20 ECMnEMK2 — ECM n Error Mask Register 2, Description added, NOTE |
| | | 2654 | 29.4.28 ECMnPE2 — ECM n Pseudo Error Trigger Register 2, Description corrected |
| | | 2655 | 29.4.29 ECMnDTMCTL — ECM n Delay Timer Control Register, Changed, Access |
| | | 2655 | 29.4.29 ECMnDTMCTL — ECM n Delay Timer Control Register, Changed, Value after reset |
| | | 2655 | 29.4.29 ECMnDTMCTL — ECM n Delay Timer Control Register, Changed, 32bit register |
| | | 2655 | Table 29.41 ECMnDTMCTL register contents, Changed, Bit Position |
| | | 2655 | Table 29.41 ECMnDTMCTL register contents, Bit Position: 1, Function |
| | | 2656 | 29.4.30 ECMnDTMR — ECM n Delay Timer Register, Description changed |
| | | 2666 | 29.5 Difference among P1M-C, P1H-C and P1H-CE, Section moved |
| | | Section 30 Data CRC Function B (DCRB) | |
| | | 2667 | Table 30.1 Units and Channels, Corrected |
| | | 2669 | Figure 30.1 HW implementation of 32-bit Ethernet CRC calculation, Changed |
| | | 2669 | Figure 30.2 HW implementation of 16-bit CCITT CRC calculation, Changed |
| | | 2670 | Figure 30.3 HW implementation of 8-bit SAE J1850 CRC calculation, Changed |
| | | 2670 | Figure 30.4 HW implementation of 8-bit 0x2F polynomial CRC calculation, Changed |
| | | 2671 | Figure 30.5 Block diagram of Data CRC Function B, Figure changed, and Note position changed |
| | | Section 32 Flash Memory | |
| | | 2683 | 32.1 Features, Description changed |
| | | 2683 | 32.1.1 Clock Supply, Description changed |
| | | 2709 | 32.11.6.1 PRDNAMEn — Product Name Storage Register (n = 1 to 4), Description changed, Value after reset |
| | | 2723 | 32.16 Reference documents, Section moved |
| | | 2723 | 32.17 Difference among P1M-C, P1H-C and P1H-CE, Section moved |
| | | Section 33 RAM Modules | |
| | | 2725 | 33.2.1 Function overview, RAM Initialization, Description changed |
| | | 2727 | 33.5 Difference among D3, D4, D5 and D5ED, Description changed |
| | | Section 34 Boundary Scan | |
| | | 2731 | Table 34.3 Pins Not Subject to Boundary Scan, Corrected |
| | | 2731 | Table 34.3 Pins Not Subject to Boundary Scan, Added, Type: Power supply pins, Pin Name: “DVCC” |
| | | 2732 | (4) IDCODE, Description changed |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | Section 35 Electrical Specifications | |
| | | 2736 | 35.2 Absolute Maximum Ratings: Description corrected (Conditions added) |
| | | 2736 | 35.2 Absolute Maximum Ratings: Corrected |
| | | 2737 | 35.2 Absolute Maximum Ratings: Note 6 added |
| | | 2737 | 35.2 Absolute Maximum Ratings, Descriptionl deleted, CAUTION |
| | | 2738 | Table 35.1 Supply Voltage Characteristics, Description changed, Item: Core supply voltage (DPS), Condition |
| | | 2738 | Table 35.1 Supply Voltage Characteristics, Added, *1, *2 |
| | | 2739 | 35.3.2 Main Oscillator Characteristics: Description corrected |
| | | 2739 | Table 35.2 Main Oscillator Characteristics, Added, *4 |
| | | 2739 | Table 35.2 Main Oscillator Characteristics, Description changed |
| | | 2740 | Figure 35.1 Oscillator Circuit Diagram (Quartz Crystal w/o External Components), Deleted |
| | | 2740 | 35.3.2 Main Oscillator Characteristics, Deleted, CAUTIONS |
| | | 2740 | Figure 35.1 Oscillator Circuit Diagram and Connection Example of Crystal Resonator, Chenged, Title and Figure |
| | | 2740 | 35.3.2 Main Oscillator Characteristics, Added, NOTE |
| | | 2740 | 35.3.3 PLL0 Characteristics: Description corrected |
| | | 2741 | 35.3.4 Internal Oscillator Characteristics: Description corrected |
| | | 2741 | Table 35.5 Operational Condition: Description corrected |
| | | 2743 | 35.4.1 Input Leakage Current, Description deleted |
| | | 2743 | 35.4.1 Input Leakage Current: Description corrected |
| | | 2744 | 35.4.2 Digital IO Pins Input and Output Characteristics: Description corrected |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics (1/3), Description changed, Input pull-up current source, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics (1/3), Description changed, Input pull-up current source, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics (1/3), Description changed, Input pull-up current source, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics (1/3), Description changed, Input pull-up current source, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics (1/3), Description changed, Input pull-down current source, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics (1/3), Description changed, Input pull-down current source, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics (1/3), Description changed, Input pull-down current source, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Added, Item: Output resistance of GPIO buffer, *8 |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Item: Output resistance of GPIO buffer, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Item: Output resistance of GPIO buffer, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Item: Output resistance of GPIO buffer, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Item: Output resistance of GPIO buffer, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Added, Item: Output resistance of GPIO buffer, *8 |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Item: Output resistance ofHSIO buffer, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Item: Output resistance ofHSIO buffer, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Item: Output resistance ofHSIO buffer, Condition |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Item: Output resistance ofHSIO buffer, Condition |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 2745 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Item: High level output voltage for HSIO buffer, Description deleted |
| | | 2745 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Item: High level output voltage for HSIO buffer |
| | | 2745 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Item: High level output voltage for HSIO buffer, Condition |
| | | 2745 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Item: High level output voltage for HSIO buffer, Condition |
| | | 2745 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Item: High level output voltage for HSIO buffer |
| | | 2745 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Item: High level output voltage for HSIO buffer |
| | | 2745 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Item: High level output voltage for HSIO buffer, Symbol |
| | | 2745 | Table 35.8Digital IO Pins: Input and Output Characteristics, Description added, Item: High level output voltage for HSIO buffer, Condition |
| | | 2745 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Item: Low level output voltage for HSIO buffer, Condition |
| | | 2745 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Item: Low level output voltage for HSIO buffer, Condition |
| | | 2745 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Item: Low level output voltage for HSIO buffer, Condition |
| | | 2745 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Item: Low level output voltage for HSIO buffer, Symbol |
| | | 2746 | Table 35.8 Digital IO Pins: Input and Output Characteristics (3/3), Description deleted, Note 1: “(TBC)” |
| | | 2746 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Note 7 |
| | | 2746 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Added, Note 8 |
| | | 2747 | 35.4.3 Analog Input Pins (ADCnIm) Characteristics: Description corrected |
| | | 2747 | Table 35.9 Analog Input Pins (ADCnIm) Characteristics, Title changed |
| | | 2747 | Table 35.9 Analog Input Pins (ADCnIm) Characteristics, Description changed, Item: Internal pull-down resistance, Condition |
| | | 2747 | Table 35.9 Analog Input Pins (ADCnIm) Characteristics, Reference changed, Note 1 |
| | | 2747 | Table 35.9 Analog Input Pins (ADCnIm) Characteristics*2: Note corrected (Note 2 deleted) |
| | | 2747 | Table 35.9 Analog Input Pins (ADCnIm) Characteristics, Added, Note 3 |
| | | 2747 | 35.4.4 Main Oscillator Input Pin (X1) Characteristics: Description corrected |
| | | 2748 | 35.4.5 Aurora Interface Clock Characteristics: Description corrected |
| | | 2748 | 35.4.6 Injected Current Characteristics: Description corrected |
| | | 2748 | Table 35.12 Injected Current Operating Conditions, Added, *2 |
| | | 2750 | 35.5.2 Power Supply Currents (D3): Description corrected |
| | | 2751 | 35.5.3 Power Supply Currents (D4): Description corrected |
| | | 2751 | 35.5.3 Power Supply Currents (D4): Description corrected |
| | | 2752 | 35.5.4 Power Supply Currents (D5): Description corrected |
| | | 2752 | 35.5.4 Power Supply Currents (D5): Description corrected |
| | | 2753 | 35.5.5 Power Supply Currents (D5ED): Description corrected |
| | | 2753 | 35.5.5 Power Supply Currents (D5ED): Description corrected |
| | | 2754 | 35.6.1.1 General Conditions: Description corrected |
| | | 2765 | Table 35.25 CSIH Timing (Master Mode, 10 MHz Communication Speed), Description changed |
| | | 2766 | Table 35.25 CSIH Timing (Master Mode, 10 MHz Communication Speed), Description changed |
| | | 2766 | Table 35.25 CSIH Timing (Master Mode, 10 MHz Communication Speed): *5 deleted and description of Note 5 corrected |
| | | 2767 | 35.6.9.3 Master mode (20 MHz Communication Speed): Description corrected |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 2767 | Table 35.26 CSIH Timing (Master Mode, SLRS = 1, 20 MHz Communication Speed): Note 5 deleted |
| | | 2767 | Table 35.26 CSIH Timing (Master Mode, SLRS = 1, 20 MHz Communication Speed): Note 1 corrected |
| | | 2768 | Table 35.27 CSIH Timing (Slave Mode, 10 MHz Communication Speed): Note 1 deleted |
| | | 2769 | 35.6.9.5 Slave mode (20 MHz Communication Speed): Description corrected |
| | | 2769 | Table 35.28 CSIH Timing (Slave Mode, 20 MHz Communication Speed): Note 1 deleted |
| | | 2770 | Figure 35.15 CSIH Master Mode Timing – SLRS = 0, Changed |
| | | 2772 | Figure 35.19 CSIH Master Mode Timing – CSS[7:0] hold (CSIHnSIT = 0): Corrected (Change color of the “S” letter) |
| | | 2772 | Figure 35.19 CSIH Master Mode Timing – CSS[7:0] hold (CSIHnSIT = 1): Corrected (Change color of the “S” letter) |
| | | 2774 | Figure 35.24 CSIH Slave Mode Timing – General Definition, Changed |
| | | 2776 | 35.6.10 FlexRay Timing: Description corrected |
| | | 2780 | Table 35.33 GTM Timing: Corrected |
| | | 2788 | Figure 35.42 LPD (4pin) Interface Timing: Corrected (Change color of the “L” letter ) |
| | | 2789 | Figure 35.43 AUDRSTZ Timing: Corrected |
| | | 2790 | Figure 35.44 AUDR AC Timing, Changed |
| | | 2796 | Table 35.50 Flash FLSCI programming characteristics, Description changed, Item: FLSCI3 transfer rate |
| | | 2796 | Table 35.50 Flash FLSCI programming characteristics: Corrected |
| | | 2796 | Table 35.50 Flash FLSCI programming characteristics, Description changed: Item |
| | | 2798 | 35.7.1 POC Characteristics: Description corrected |
| | | 2798 | 35.7.1 POC Characteristics: Description corrected |
| | | 2799 | 35.7.2 Core Voltage Monitor (CVM) Characteristics: Description corrected |
| | | 2799 | 35.7.2 Core Voltage Monitor (CVM) Characteristics: Description corrected |
| | | 2800 | 35.7.3 A/D Converter Characteristics: Description corrected |
| | | 2800 | 35.7.3 A/D Converter Characteristics: Description corrected |
| | | 2800 | Table 35.54 ADC Characteristics, Description changed, Item: Self-diagnosis accuracy, Condition |
| | | 2800 | Table 35.54 ADC Characteristics, Description changed, Item: Self-diagnosis accuracy, Condition |
| | | 2800 | Table 35.54 ADC Characteristics, Added, Item: TDE switch resistance, Condition |
| | | 2800 | Table 35.54 ADC Characteristics, Added, Note 2, Note 3, Symbol: RIN1, CIN1 |
| | | 2801 | Figure 35.55 Equivalent circuit, Added |
| | | 2802 | 35.7.4 Temperature Sensor Characteristics: Description corrected |
| | | 2802 | 35.7.4 Temperature Sensor Characteristics: Description corrected |
| | | 2802 | 35.7.4 Temperature Sensor Characteristics: Description corrected |
| | | 2802 | Table 35.55 Temperature Sensor (TS) Characteristics: Corrected |
| | | 2803 | Table 35.57 Programming Characteristics, Description added |
| | | 2803 | Table 35.57 Programming Characteristics, Description added |
| | | 2803 | Table 35.57 Programming Characteristics, Description added |
| | | 2803 | Table 35.57 Programming Characteristics, Description added |
| | | 2803 | Table 35.57 Programming Characteristics, Description added |
| | | 2803 | Table 35.57 Programming Characteristics, Description added |
| | | 2803 | Table 35.57 Programming Characteristics, Added, Note 1 |
| | | 2808 | Table 35.62 Thermal Characteristics - Junction to Board Resistance (Ψjb), Added, Symbol: *1 |
| | | 2808 | Table 35.62 Thermal Characteristics - Junction to Board Resistance (Ψjb), Added, Condition: *2*3 |
| | | 2808 | Table 35.62 Thermal Characteristics - Junction to Board Resistance (Ψjb): Corrected (MAX value) |
| | | 2808 | Table 35.62 Thermal Characteristics - Junction to Board Resistance (Ψjb): Note 1 corrected |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.10 | Jul 15, 2016 | 2808 | Table 35.62 Thermal Characteristics - Junction to Board Resistance (Ψjb), Description changed, Note 3 |
| | | Section 36 Package | |
| | | 2809 | 36.1 Package Line Up, Changed, RH850/P1x-C Product: Name, P1M-C 144 |
| | | Section 37 CAN Controller (MCAN) | |
| | | 2814 | Section 37 CAN Controller (MCAN), Description changed |
| | | 2815 | Table 37.2 Number of MTTCAN/MCAN channel, Deleted, Macro: MCAN0, *2 |
| | | 2815 | Table 37.2 Number of MTTCAN/MCAN channel, Changed, Macro: MCAN0, Description |
| | | 2815 | Table 37.2 Number of MTTCAN/MCAN channel, Deleted, Note 2 |
| | | 2816 | Table 37.4 Clock Supply, Description changed, Note 2 |
| | | 2819 | 37.2.1 Functional overview, Description deleted |
| | | 2820 | Figure 37.1 MCAN/MTTCAN block diagram, Changed |
| | | 2820 | 37.2.2 Block Diagram, Description deleted |
| | | 2820 | 37.2.2 Block Diagram, Description deleted, Note 1 |
| | | 2822 | 37.4.1 Procedure of Module Standby and Limited Reset: Description corrected |
| | | 2822 | 37.4.1 Procedure of Module Standby and Limited Reset: Description corrected |
| | | 2827 | Table 37.14 M_CAN Register Map: Corrected |
| | | 2880 | Figure 37.4 Message RAM Configuration: Corrected |
| | | 2894 | (4) Transmitter Delay Compensation: Titile corrected |
| | | 2895 | Figure 37.5 Transmitter delay measurement: Title corrected |
| | | 2915 | Figure 37.14 M_TTCAN Block Diagram: Title corrected |
| | | All | Description changed, “M_CAN” → “M_TTCAN” |
| | | 2993 | Figure 37.15 Message RAM Configuration: Corrected |
| | | All | Description changed, “m_can” → “m_ttcan” |
| | | 3009 | (4) Transceiver Delay Compensation: Description corrected |
| | | 3010 | Figure 37.16 Transmitter delay measurement: Title corrected |
| | | 3054 | 37.7 Usage Notes, Section moved |
| | | 3055 | 37.8 CAUTION, Section moved |
| | | 3061 | 37.9 Difference among P1M-C, P1H-C and P1H-CE, Section moved |
| 1.20 | Jan 12, 2017 | 4 | How to Use This Manual, Changed, Organization |
| | | 4 | How to Use This Manual, Description added, Organization |
| | | 4 | How to Use This Manual, Description added, How to read this manual |
| | | Section 1 Overview | |
| | | 49 | 1.4 Features, Description deleted, FlexRay |
| | | Section 2 Pin Functions | |
| | | 58 | 2.1.1 P1M-C (QFP-144) (TOP View), Changed, Pin 144 |
| | | 58 | 2.1.1 P1M-C (QFP-144) (TOP View), Description deleted, /VCC |
| | | 79 | Table 2.3 Pin Function assignments, Deleted, No.224: MEMC0CLK |
| | | 81 | Table 2.4 Pin Function Name Definition, "n" added, No.31: CSIH0DCS |
| | | 82 | Table 2.4 Pin Function Name Definition, "n" added, No.47: CSIH1DCS |
| | | 82 | Table 2.4 Pin Function Name Definition, "n" added, No.63: CSIH2DCS |
| | | 82 | Table 2.4 Pin Function Name Definition, "n" added, No.79: CSIH3DCS |
| | | 87 | Table 2.4 Pin Function Name Definition, Deleted, No. 255: MEMC0CLK |
| | | 87 | Table 2.4 Pin Function Name Definition, Line number changed |
| | | 88 | Table 2.4 Pin Function Name Definition, Description deleted, Explanation: Flash Writer I/F |
| | | Section 3 CPU System | |
| | | 159 | 3.1.1 Block Configuration, Description changed, ICUMC |
| | | 179 | Table 3.25 PID Register Contents, Description added, Value after reset: Bits 23 to 8 |
| | | 220 | Table 3.77 SEGCONT register contents, Description changed, Function: Bit 2 |
| | | 221 | Table 3.78 SEGFLAG register contents, Description changed, Function: Bit 5 |
| | | 249 | 3.4.8 System Register Hazards, Description added, Load/Store |
| | | Section 4 Address Map | |
| | | 253 | Figure 4.1 Address Map (P1M-C, P1H-C and P1H-CE), Description added |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.20 | Jan 12, 2017 | 267 | Table 4.1 List of P-bus area access, Corrected, Note 2 |
| | | 268 | Table 4.2 List of H-Bus area access, Description changed, Table header |
| | | Section 6 Interrupt Controller (INTC) | |
| | | 291 | 6.4 Interrupt Controller Control Registers, Changed |
| | | 323 | Table 6.25 Interrupt latency, Changed, Latency (PE1/2): INTC2 |
| | | Section 7 DMA | |
| | | 331 | 7.1.2 Clock Supply, Description changed |
| | | Section 8 Reset Controller | |
| | | 524 | 8.4.7 Start Up of Cores, Description changed |
| | | Section 16 Clocked Serial Interface H (CSIH) | |
| | | 681 | Table 16.29 CSIHnCFGx register contents, Figures in Table changed, Clock and data phase selection |
| | | 681 | Table 16.29 CSIHnCFGx register contents, Figures in Table changed, Clock and data phase selection |
| | | 684 | 16.3.16 CSIHnBRSx — CSIH baud rate setting register, "n" deleted, Section title |
| | | 686 | 16.3.18 PMMAnTCTLx — PMM TG control register, Changed, Bits 23 to 16 |
| | | 688 | 16.3.19 PMMAnTTXm — PMM TG transmit buffer, "A" deleted, Section title |
| | | 689 | 16.3.20 PMMAnRX — PMM TG receive buffer, "A" deleted, Section title |
| | | 690 | 16.3.21 PMMAnTCNTx — PMM TGx counter, "A" deleted, Section title |
| | | 691 | 16.3.22 PMMAnSTR0 — PMM TG status register 0, "A" deleted, Section title |
| | | 692 | 16.3.23 PMMAnSTR1 — PMM TG status register 1, "A" deleted, Section title |
| | | 693 | 16.3.24 PMMAnSTC — PMM TG status clear register, "A" deleted, Section title |
| | | Section 17 High Speed USRT (HS-USRT) | |
| | | 796 | Figure 17.1 Block Diagram of HS-USRT, Corrected, De-sirializer |
| | | Section 18 LIN/UART Interface (RLIN3) | |
| | | 908 | 18.3.4.6 RLN3nLEDE — UART Error Detection Enable Register, Description changed, FERE/OERE/BERE Bit |
| | | 926 | Table 18.71 Interrupt Sources, Description changed, RLIN3n Successful Reception Interrupt: LIN master mode |
| | | 979 | 18.5.6.1 LIN Master Mode, Description deleted, NOTE |
| | | 979 | 18.5.6.2 LIN Slave Mode, Description changed, NOTE |
| | | 979 | 18.5.6.2 LIN Slave Mode, Description deleted, NOTE |
| | | 980 | 18.5.6.3 UART Mode, Changed, NOTE |
| | | 980 | 18.5.6.3 UART Mode, Description deleted, NOTE |
| | | Section 19 CAN Controller (MCAN) | |
| | | 997 | Table 19.12 M_CAN Register Map, Changed, RESET: MCANnCREL |
| | | 998 | Table 19.12 M_CAN Register Map, Description deleted, Note |
| | | 999 | 19.5.2.3 (1) MCANnCREL — Core Release Register, Changed, Value after reset |
| | | 999 | 19.5.2.3 (1) MCANnCREL — Core Release Register, Changed, Value after reset: Bits 31 to 0 |
| | | 999 | Table 19.14 Coding of Revisions, Description deleted, Table title |
| | | 999 | Table 19.14 Coding of Revisions, Description changed |
| | | 1001 | Section 19, Changed, "Transceiver delay compensation" to "Transmitter delay compensation" |
| | | 1005 | Section 19, Changed, "ISO11898-1" to "ISO11898-1:2015" |
| | | 1005 | Table 19.19 MCANnCCCR Register Contents, NOTE deleted, Function: Bit 15 |
| | | 1094 | Table 19.71 M_TTCAN Register Map, Changed, RESET: MTTCAN0CREL |
| | | 1095 | Table 19.71 M_TTCAN Register Map, Description deleted, Note |
| | | 1096 | 19.6.2.3 (1) MTTCAN0CREL — Core Release Register, Changed, Value after reset |
| | | 1096 | 19.6.2.3 (1) MTTCAN0CREL — Core Release Register, Changed, Value after reset: Bits 31 to 0 |
| | | 1096 | Table 19.73 Coding of Revisions, Description deleted, Table title |
| | | 1096 | Table 19.73 Coding of Revisions, Description changed |
| | | 1102 | Table 19.78 MTTCAN0CCCR Register Contents, NOTE deleted, Function: Bit 15 |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.20 | Jan 12, 2017 | Section 20 FlexRay (FLX) | |
| | | 1239 | Section 20 FlexRay (FLX), Description deleted |
| | | 1242 | 20.1.7 Functions, Description in Table deleted, Specification: Communication |
| | | 1531 | 20.5 Difference among P1M-C, P1H-C and P1H-CE, Deleted |
| | | Section 21 Ethernet Controller (ETNA) | |
| | | 1655 | Figure 21.12 Sample Transmission Flowchart (Single-Frame/Two-Description in store and forward mode), Figure changed |
| | | Section 22 Single Edge Nibble Transmission (SENT) | |
| | | 1682 | 22.2 Overview, Description changed, Variable data transmission rate |
| | | 1682 | 22.2 Overview, Changed, Overview of Functions |
| | | 1690 | 22.3.3 RSENTnCC — RSENT Communication Configuration Register, Description changed, RSENTnCC.SMF (Serial Message Format) |
| | | 1742 | 22.4.3.2 (5) Slow Channel Message Reception, Description changed |
| | | Section 23 Window Watchdog Timer A (WDTA) | |
| | | 1762 | Table 23.17 WDTA start-up options, Added, Description: OPWD0MD |
| | | Section 25 Generic Timer Module (GTM) | |
| | | 1807 | Figure 25.1 GTM integration diagram, Description changed |
| | | 2233 | 25.21 CAUTION, Description on Table header deleted, "R7F701327" |
| | | 2233 | 25.21 CAUTION, ID13 and ID14 added |
| | | 2236 | 25.21 CAUTION, ID13 and ID14 added |
| | | Section 27 A/D Converter (ADCF) | |
| | | 2272 | Table 27.11 ADCFnVCRj register contents, Changed, Function: Bit 7 |
| | | 2286 | 27.3.16 ADCFnULLMTBR0 to 2 — Upper-Limit/Lower-Limit Table Registers 0 to 2, Description changed, Access |
| | | 2305 | Figure 27.4 Example of Operation in Multicycle Scan Mode, Description added |
| | | 2306 | Figure 27.5 Example of Operation in Continuous Scan Mode, Description added |
| | | 2307 | Figure 27.6 Example of Normal A/D Conversion Operation in Addition Mode, Description added |
| | | 2308 | Figure 27.7 Example of Using an External Analog Multiplexer (Port Output), Description changed |
| | | 2308 | Figure 27.7 Example of Using an External Analog Multiplexer (Port Output), Note added |
| | | 2319 | 27.4.15 Scan End Interrupt Request, Changed |
| | | 2320 | Figure 27.17 Example of an MPX Interrupt Occurrence, Description changed |
| | | 2323 | Figure 27.20 Trigger Input Flow (SG0 to SG2), Description changed |
| | | 2324 | Figure 27.21 Trigger Input Flow (SG3 to SG4), Figure changed |
| | | 2327 | 27.7 Difference among P1M-C, P1H-C and P1H-CE, Description changed |
| | | 2327 | Table 27.44 Different specifications of ADCF, Changed, Table header |
| | | 2327 | Table 27.44 Different specifications of ADCF, Corrected, ADCFn module |
| | | Section 28 Functional Safety | |
| | | 2329 | 28.1 Overview, Description changed |
| | | 2344 | 28.2.2.3 (11) CFSTSTCTL_VCI2CFBA/PE1/PE2 — FLI (Code-Flash) Sub-Test Control Register, Description added |
| | | 2354 | Table 28.24 DFEADRn register contents, Changed, Function: Bits 20 and 19 |
| | | 2440 | Table 28.89 List of Registers, *2 added, Symbol: VPECCCTL_SGn |
| | | 2442 | Table 28.89 List of Registers, *2 added, Symbol: VPECCCTL_TERM_SG7 |
| | | 2442 | Table 28.89 List of Registers, *2 added, Symbol: VPECCCTL_VCI2VPI |
| | | 2445 | Table 28.89 List of Registers, Note 2 added |
| | | 2458 | 28.2.9.5(13) LSSECCCTL_PE1/PE2 — LSU Slave Data and Address ECC Control Register, Changed, Bits 3 and 2 |
| | | 2458 | 28.2.9.5(13) LSSECCCTL_PE1/PE2 — LSU Slave Data and Address ECC Control Register, Changed, R/W: Bits 3 and 2 |
| | | 2458 | Table 28.102 LSSECCCTL_PE1/PE2 register contents, Bits 3 and 2 deleted |
| | | 2458 | Table 28.102 LSSECCCTL_PE1/PE2 register contents, Description added, Function: bit 1 |
| | | 2458 | Table 28.102 LSSECCCTL_PE1/PE2 register contents, Description added, Function: bit 0 |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.20 | Jan 12, 2017 | 2470 | 28.2.9.5 (25) VPECCCTL_SGn — System Interconnect (PBus I/F) Data ECC Control Register, NOTE added |
| | | 2476 | 28.2.9.5 (31) VPECCCTL_TERM_SG7 — PBus Data and Address ECC Control Register, NOTE added |
| | | 2482 | 28.2.9.5 (37) VPECCCTL_VCI2VPI — System Interconnect Address ECC Control Register, NOTE added |
| | | 2524 | 28.4.1 Overview, Description changed |
| | | 2527 | 28.4.1.3 Global RAM protection, Changed, Register names |
| | | 2528 | 28.4.1.3 Global RAM protection, Description changed |
| | | 2530 | Table 28.161 List of Registers, Column added, PBG |
| | | Section 29 Error Control Module (ECM) | |
| | | 2608 | Table 29.8 List of Error Inputs, Added, Error source explanation: No.22 |
| | | 2611 | Table 29.8 List of Error Inputs, Added, Error source explanation: No.37 |
| | | 2612 | Table 29.8 List of Error Inputs, Added, Error source explanation: No.54 |
| | | 2625 | 29.4.3 ECMmnECLR — ECM Master/Checker n Error Clear Trigger Register, Changed, CAUTIONS |
| | | 2641 | 29.4.18 ECMnEMK0 — ECM n Error Mask Register 0, Changed, NOTE |
| | | 2642 | 29.4.19 ECMnEMK1 — ECM n Error Mask Register 1, Changed, NOTE |
| | | 2643 | 29.4.20 ECMnEMK2 — ECM n Error Mask Register 2, Changed, NOTE |
| | | Section 31 On-chip Debug Unit (OCD) | |
| | | 2677 | 31.1 (17) Security Function, Description changed |
| | | Section 32 Flash Memory | |
| | | 2684 | Table 32.4 List of Registers, Changed, Access Protection |
| | | 2684 | Table 32.4 List of Registers, Note 1 added |
| | | 2696 | Table 32.14 List of Enabling of Self-Programming Register, Columns added |
| | | 2696 | Table 32.14 List of Enabling of Self-Programming Register, Note 1 added |
| | | 2703 | 32.11.4.1 FHVE15 — FHVE15 Control Register, Description added |
| | | 2703 | Table 32.19 FHVE15 register contents, Description added, Function: Bit 0 |
| | | 2704 | 32.11.4.2 FHVE3 — FHVE3 Control Register, Description added |
| | | 2704 | Table 32.20 FHVE3 register contents, Description added, Function: Bit 0 |
| | | 2708 | Table 32.23 OPBT0 register contents, Added, Function: Bit 24 |
| | | 2711 | 32.12.3 OPBT2 — Option Byte 2, Description changed |
| | | 2712 | 32.12.4 OPBT13 — Option Byte 13, Description changed |
| | | 2718 | 32.15 Usage Notes, Description changed, 4. |
| | | 2718 | 32.15 Usage Notes, Description added, 4. Resets during programming and erasure |
| | | 2718 | 32.15 Usage Notes, Description changed, 8. |
| | | Section 33 RAM Modules | |
| | | 2724 | 33.4 Usage Notes, Description added |
| | | Section 34 Boundary Scan | |
| | | 2728 | Table 34.3 Pins Not Subject to Boundary Scan, Changed, Pin Name: "." → "," |
| | | Section 35 Electrical Specifications | |
| | | 2734 | 35.2 Absolute Maximum Ratings, Corrected, Item: Tstg |
| | | 2734 | 35.2 Absolute Maximum Ratings, Changed, Note 3 |
| | | 2741 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Condition: Ro11 |
| | | 2741 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Condition: Ro12 |
| | | 2741 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Condition: Ro13 |
| | | 2741 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Condition: Ro14 |
| | | 2742 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Changed, Condition: Ro21 |
| | | 2742 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Condition: Ro21 |
| | | 2742 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Changed, Condition: Ro22 |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.20 | Jan 12, 2017 | 2742 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Condition: Ro22 |
| | | 2742 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Changed, Condition: Ro23 |
| | | 2742 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Condition: Ro23 |
| | | 2742 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Changed, Condition: Ro24 |
| | | 2742 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description added, Condition: Ro24 |
| | | 2744 | Table 35.8 Digital IO Pins: Input and Output Characteristics, Description changed, Notes 2 and 3 |
| | | 2745 | Table 35.9 Analog Input Pins (ADCnIm) Characteristics*3, Changed: *2 → *3 |
| | | 2745 | Table 35.9 Analog Input Pins (ADCnIm) Characteristics*3, Added, Condition |
| | | 2745 | Table 35.9 Analog Input Pins (ADCnIm) Characteristics*3, Added, Condition |
| | | 2745 | Table 35.9 Analog Input Pins (ADCnIm) Characteristics*3, Added, Condition |
| | | 2745 | Table 35.9 Analog Input Pins (ADCnIm) Characteristics*3, Added, Condition |
| | | 2745 | Table 35.9 Analog Input Pins (ADCnIm) Characteristics, Added, Note 2 |
| | | 2748 | Table 35.13 Supply Current Consumption: Device P1M-C, *1 deleted, Table title |
| | | 2748 | Table 35.13 Supply Current Consumption: Device P1M-C, Note 1 deleted |
| | | 2748 | Table 35.13 Supply Current Consumption: Device P1M-C, Note 2 added |
| | | 2749 | Table 35.14 Supply Current Consumption: Device P1H-C (4MB), *1 deleted, Table title |
| | | 2749 | Table 35.14 Supply Current Consumption: Device P1H-C (4MB), Note 1 deleted |
| | | 2749 | Table 35.14 Supply Current Consumption: Device P1H-C (4MB), Note 2 added |
| | | 2750 | Table 35.15 Supply Current Consumption: Device P1H-C (8MB), *1 deleted, Table title |
| | | 2750 | Table 35.15 Supply Current Consumption: Device P1H-C (8MB), Note 1 deleted |
| | | 2750 | Table 35.15 Supply Current Consumption: Device P1H-C (8MB), Note 2 added |
| | | 2751 | Table 35.16 Supply Current Consumption: Device P1H-CE, *1 deleted, Table title |
| | | 2751 | Table 35.16 Supply Current Consumption: Device P1H-CE, Deleted, Note 1 |
| | | 2751 | Table 35.16 Supply Current Consumption: Device P1H-CE, Note 2 added |
| | | 2755 | 35.6.2.1 Power Up sequencing, Section added |
| | | 2780 | Figure 35.36 MEMC0 – READ Access Timing, Description changed |
| | | 2783 | Figure 35.37 MEMC0 – Write Access Timing, Description changed |
| | | 2794 | 35.6.24 Flash Programming Characteristics, Description changed, Conditions: |
| | | 2794 | Table 35.51 Flash FLSCI programming characteristics, Description changed, Item |
| | | 2794 | Table 35.51 Flash FLSCI programming characteristics, Description changed, Item |
| | | 2794 | Table 35.51 Flash FLSCI programming characteristics, Description changed, Item |
| | | 2794 | Figure 35.52 FLSCI Timing, Description changed |
| | | 2798 | Table 35.55 ADC Characteristics, Rows added |
| | | 2798 | Table 35.55 ADC Characteristics, Deleted, Condition: INL, OSE, FSE and QNE |
| | | 2798 | Table 35.55 ADC Characteristics, Changed, MIN./MAX.: OSE and FSE |
| | | 2801 | Table 35.57 Basic Characteristics, Description added, Note 2 |
| | | 2803 | Table 35.60 Basic Characteristics, Description added, Note 2 |
| | | 2803 | Table 35.60 Basic Characteristics, Changed, Note 1 |
| | | 2806 | Table 35.63 Thermal Characteristics - Junction to Board Resistance, Description deleted, Condition |
| | | Section 37 CAN Controller (MCAN) | |
| | | 2825 | Table 37.12 M_CAN Register Map, Changed, RESET: MCANnCREL |
| | | 2826 | Table 37.12 M_CAN Register Map, Description deleted, Note |
| | | 2827 | 37.5.2.3 (1) MCANnCREL — Core Release Register, Changed, Value after reset |
| | | 2827 | 37.5.2.3 (1) MCANnCREL — Core Release Register, Changed, Value after reset: Bits 31 to 0 |
| | | 2827 | Table 37.14 Coding of Revisions, Description deleted, Table title |
| | | 2827 | Table 37.14 Coding of Revisions, Description changed |
| | | 2829 | Section 37, Changed, "Transceiver delay compensation" to "Transmitter delay compensation" |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.20 | Jan 12, 2017 | 2916 | Table 37.68 M_TTCAN Register Map, Changed, RESET: MTTCAN0CREL |
| | | 2917 | Table 37.68 M_TTCAN Register Map, Description deleted, Note |
| | | 2918 | 37.6.2.3 (1) MTTCAN0CREL — Core Release Register, Changed, Value after reset |
| | | 2918 | 37.6.2.3 (1) MTTCAN0CREL — Core Release Register, Changed, Value after reset: Bits 31 to 0 |
| | | 2918 | Table 37.70 Coding of Revisions, Description deleted, Table title |
| | | 2918 | Table 37.70 Coding of Revisions, Description changed |
| 1.21 | Mar 10, 2017 | Section 1 Overview | |
| | | 40 | Table 1.1 Product list and package, Rows added, "R7F701396A" and "R7F701397A" |
| | | 40 | 1.2 Product List, Columns for BGA-156 added |
| | | 41 | Table 1.2 Features in each product, Description changed, P1H-C (4MB) |
| | | 41 | Table 1.2 Features in each product, Description changed, P1M-C |
| | | Section 2 Pin Functions | |
| | | 61 | 2.1.6 P1M-C/P1H-C (BGA-156) (TOP View), Section added |
| | | 61 | 2.1.7 Term Definition, Column for P1M-C/P1H-C (156) added |
| | | 61 | 2.1.7 Term Definition, Description in Table added, Table header |
| | | 63 | Table 2.1 Pin List, Column added |
| | | 75 | Table 2.3 Pin Function assignments, Columns added |
| | | 75 | Table 2.3 Pin Function assignments, Description in the table header added |
| | | Section 4 Address Map | |
| | | 257 | Table 4.1 List of P-bus area access, Description added, Table header |
| | | 257 | Table 4.1 List of P-bus area access, Columns added |
| | | 267 | Table 4.1 List of P-bus area access, Note 3 added |
| | | 268 | Table 4.2 List of H-Bus area access, Description added, Table header |
| | | 268 | Table 4.2 List of H-Bus area access, Columns added |
| | | 268 | Table 4.2 List of H-Bus area access, Note 1 added |
| | | Section 6 Interrupt Controller (INTC) | |
| | | 327 | Table 6.26 Difference among P1M-C, P1H-C and P1H-CE, *3 added |
| | | 330 | Table 6.26 Difference among P1M-C, P1H-C and P1H-CE, Note 3 added |
| | | Section 7 DMA | |
| | | 374 | Table 7.12 List of DMA Trigger Sources, Description added, DMA Trigger Source: DMACTRG[8] and DMACTRG[9] |
| | | 376 | Table 7.12 List of DMA Trigger Sources, Description added, DMA Trigger Source: DMACTRG[94] to DMACTRG[97] |
| | | 376 | Table 7.12 List of DMA Trigger Sources, Description added, DMA Trigger Source: DMACTRG[102] to DMACTRG[105] |
| | | 378 | Table 7.13 List of DTS Trigger Sources, Description added, Primary DTS Trigger Source: DTSTRG[8] and DTSTRG[9] |
| | | 381 | Table 7.13 List of DTS Trigger Sources, Description added, Primary DTS Trigger Source: DTSTRG[104] to DTSTRG[107] |
| | | 381 | Table 7.13 List of DTS Trigger Sources, Description added, Primary DTS Trigger Source: DTSTRG[112] to DTSTRG[115] |
| | | Section 15 External Memory Controller (MEMC) | |
| | | 645 | Table 15.12 Difference of number of macro, Description added, Table header |
| | | 645 | Table 15.12 Difference of number of macro, Columns for BGA-156 added, P1M-C/P1H-C |
| | | Section 16 Clocked Serial Interface H (CSIH) | |
| | | 650 | Table 16.10 Data Consistency Checking and Port Pins, Description added, Data Consistency Checking: CSIH0 and CSIH1 |
| | | 650 | Table 16.10 Data Consistency Checking and Port Pins, Description changed, Data Consistency Checking: CSIH3 |
| | | 790 | 16.8 Difference among P1M-C, P1H-C and P1H-CE, Description added |
| | | Section 17 High Speed USRT (HS-USRT) | |
| | | 791 | Table 17.1 HS-USRT Channels, Description added, Product Name |
| | | 839 | 17.5 Difference among P1M-C, P1H-C and P1H-CE, Description added |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.21 | Mar 10, 2017 | Section 18 LIN/UART Interface (RLIN3) | |
| | | 840 | Table 18.1 Units, Description changed, Table header |
| | | 840 | Table 18.1 Units, Columns added |
| | | 840 | Table 18.2 Unit Configurations and Channels, Description added, Table header |
| | | 840 | Table 18.2 Unit Configurations and Channels, Columns added |
| | | 983 | Table 18.95 Different specification of RLIN3n, Description added, Table header |
| | | Section 19 CAN Controller (MCAN) | |
| | | 984 | Section 19 CAN Controller (MCAN), Description added |
| | | 985 | Table 19.2 Number of MTTCAN/MCAN channel, *1 added, P1H-C (4MB): MTTCAN0 |
| | | 985 | Table 19.2 Number of MTTCAN/MCAN channel, Description changed, Note 1 |
| | | 990 | 19.2.2 Block Diagram, Note number changed |
| | | 990 | 19.2.2 Block Diagram, Note 1 added |
| | | Section 20 FlexRay (FLX) | |
| | | 1239 | Table 20.1 Number of FlexRay Modules, Description added, Table header |
| | | 1531 | 20.5 Difference among P1M-C, P1H-C and P1H-CE, Description added, Table header |
| | | Section 21 Ethernet Controller (ETNA) | |
| | | 1532 | Table 21.1 Ethernet Channels, Description added, Table header |
| | | 1534 | Table 21.5 Difference of number of macro, Column added |
| | | 1534 | Table 21.5 Difference of number of macro, Description added, Table header |
| | | 1534 | Table 21.6 Ethernet External Input / Output Signals, Note 3 added |
| | | 1553 | Table 21.15 ETNAnECSRm register contents, Note added |
| | | 1556 | Table 21.18 ETNAnPLSRm register contents, Description added, Function: Bit 0 |
| | | 1556 | Table 21.18 ETNAnPLSRm register contents, Note added |
| | | Section 22 Single Edge Nibble Transmission (SENT) | |
| | | 1678 | Table 22.2 The number of channels in each product, Description added, Table header |
| | | 1678 | Table 22.2 The number of channels in each product, Columns added |
| | | 1678 | Table 22.3 Register Base Address <RSENTn_base>, Columns added |
| | | 1678 | Table 22.3 Register Base Address <RSENTn_base>, Description deleted, <RSENTn_base> Address: RSENT6 to RSENT9 |
| | | 1684 | Table 22.8 Overview of RSENTn Registers, Description deleted, Register Name: RSENT6TSSEL to RSENT9TSSEL |
| | | 1752 | Table 22.43 Different specification of RSENT, Description added, Table header |
| | | 1752 | Table 22.43 Different specification of RSENT, Columns added |
| | | Section 25 Generic Timer Module (GTM) | |
| | | 1801 | Table 25.2 Sub-Units and Channels, *2 and *3 added, Instance name (P1H-C, P1H-CE): ATOM |
| | | 1801 | Table 25.2 Sub-Units and Channels, Notes 2 and 3 added |
| | | 1805 | Table 25.6 Pin Function Information, Description added, Table header |
| | | Section 26 Peripheral Interconnect (PIC) | |
| | | 2239 | Table 26.4 Pin Function Information, Description added, Table header |
| | | 2256 | Table 26.15 Connections between RLIN3 and GTM, Description added, Table header |
| | | 2258 | Table 26.16 Each device different specification, Note 1 and *1 added |
| | | Section 27 A/D Converter (ADCF) | |
| | | 2327 | Table 27.44 Different specifications of ADCF, Description added, Table header |
| | | 2327 | Table 27.44 Different specifications of ADCF, Columns added |
| | | Section 28 Functional Safety | |
| | | 2542 | Table 28.169 PBG Group, Description added, Note: PBG channel numbers 8 and 9 (PBG1 #1) |
| | | 2544 | Table 28.169 PBG Group, Description added, Note: PBG channel numbers 8 and 9 (PBG3 #1) |
| | | 2550 | Table 28.170 List of Registers, Description added, Note: FFEE0240H to FFEE024CH |
| | | 2554 | Table 28.170 List of Registers, Description added, Note: FFF94240H to FFF9424CH |
| | | 2571 | Table 28.184 List of Registers, Description added, Note: FFFA0020H to FFFA003CH |
| | | 2572 | Table 28.184 List of Registers, Description added, Note: FFFA0060H to FFFA00DCH |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.21 | Mar 10, 2017 | Section 29 Error Control Module (ECM) | |
| | | 2606 | Table 29.8 List of Error Inputs, Description added, Table header |
| | | 2606 | Table 29.8 List of Error Inputs, Column added, Product |
| | | 2664 | 29.5 Difference among P1M-C, P1H-C and P1H-CE, Description added |
| | | Section 32 Flash Memory | |
| | | 2707 | 32.11.5.1 PRDNAMEn — Product Name Storage Register, Description in Table headers added, Value after reset |
| | | 2707 | 32.11.5.1 PRDNAMEn — Product Name Storage Register, Table added, Value after reset |
| | | Section 34 Boundary Scan | |
| | | 2731 | Table 34.5 ID Register Codes, Rows added |
| | | 2731 | Table 34.5 ID Register Codes, Description added, Products: R7F701372AEABG |
| | | Section 35 Electrical Specifications | |
| | | 2739 | Table 35.5 Operational Condition, Description added, Condition: Junction temperature |
| | | 2739 | Table 35.5 Operational Condition, Row added, Junction temperature |
| | | 2763 | Table 35.25 CSIH Communication Speed Overview, Description changed, Table header |
| | | 2763 | Table 35.25 CSIH Communication Speed Overview, Columns added |
| | | 2806 | Table 35.63 Thermal Characteristics - Junction to Board Resistance (Ψjb), Row added, P1M-C |
| | | 2806 | Table 35.63 Thermal Characteristics - Junction to Board Resistance (Ψjb), Row added, P1H-C (4MB) |
| | | Section 36 Package | |
| | | 2808 | 36.1 Package Line Up, Column added |
| | | 2808 | 36.1 Package Line Up, Description added, Table header |
| | | 2810 | 36.2.2 FBGA (156pin) Package Drawing, Section added |
| 1.30 | Dec 25, 2017 | 3 | Notes for CMOS Devices, Corrected, Title header |
| | | 3 | Notes for CMOS Devices, Description changed, (2) Handling of unused input pins |
| | | 5 | Description of Registers, Description changed |
| | | 7 | Table of Contents, Corrected, Title header |
| | | Section 2 Pin Functions | |
| | | 76 | Table 2.3 Pin Function assignments, Description added, P1M-C/P1H-C(BGA-156) (2nd Alternative): FLXNTUOUT |
| | | 76 | Table 2.3 Pin Function assignments, Description deleted, P1M-C/P1H-C (BGA-156) (Special Function): ICUMGPIO0 and ICUMGPIO1 |
| | | 99 | 2.3 Port Overview, Corrected, Section title |
| | | 103 | Figure 2.8 Port Control Logic Block Diagram, Description in the figure changed |
| | | Section 3 CPU System | |
| | | 214 | Table 3.68 Register Contents of IPGECRUM, Description changed, Function: Bit 0 |
| | | 251 | 3.4.9 Restrictions to use DMAC0/1, DTS ICUM, Section added |
| | | 251 | 3.4.10 Restrictions to use HBU Master (Ethernet, HS-USRT, FlexRay), Section added |
| | | Section 4 Address Map | |
| | | 256 | 4.2 Address space viewed from each bus master, Corrected, Section title |
| | | 268 | Table 4.1 List of P-bus area access, Description changed, FFFF $5000_H$ to FFFF $7AFF_H$ |
| | | Section 6 Interrupt Controller (INTC) | |
| | | 293 | 6.4.1 EICn — EI level interrupt control registers, CAUTION added |
| | | 294 | 6.4.2 IMRm — EI level interrupt mask registers, CAUTION 2 added |
| | | 322 | 6.10 Broadcast notification function processor elements, Description deleted |
| | | 322 | 6.10 Broadcast notification function processor elements, Description deleted |
| | | 323 | Table 6.23 Corresponding register setting (INTC2), Description changed, Content/Description: IMK |
| | | Section 12 Clock Controller | |
| | | 571 | Figure 12.1 Clock diagram (user's perspective), Description in the figure deleted |
| | | Section 18 LIN/UART Interface (RLIN3) | |
| | | 876 | 18.3.2.19 RLN3nLDBRb — LIN Data Buffer b Register, Description changed, For response reception |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.30 | Dec 25, 2017 | 901 | 18.3.3.18 RLN3nLDBRb — LIN Data Buffer b Register, Description changed, For response reception |
| | | 955 | Figure 18.20 Target Time Area for LIN Error Detection (LIN Slave Mode), Description in the figure changed |
| | | Section 19 CAN Controller (MCAN) | |
| | | 1002 | Table 19.16 MCANnDBTP Register Contents, Description added, Bits 20 to 16 |
| | | 1099 | Table 19.75 MTTCAN0DBTP Register Contents, Description added, Bits 20 to 16 |
| | | 1166 | 19.6.2.3 (59) MTTCAN0TURNA — TUR Numerator Actual, Description changed, Value after reset |
| | | 1166 | 19.6.2.3 (59) MTTCAN0TURNA — TUR Numerator Actual, Description changed, Value after reset: Bit 16 |
| | | 1237 | 19.8 CAUTION, Description in the table header deleted, "R7F701327" |
| | | 1237 | 19.8 CAUTION, ID2 added |
| | | 1238 | 19.8 CAUTION, Description changed, "CCCR.EFBI" to "M(TT)CANnCCCR.EFBI" |
| | | 1238 | 19.8 CAUTION, Description changed, "M_CAN" to "M(TT)CAN" |
| | | 1238 | 19.8 CAUTION, Description changed, "m_can_rx" to "m(tt)can_rx" |
| | | 1238 | 19.8 CAUTION, Description changed, "CCCR.INIT" to "M(TT)CANnCCCR.INIT" |
| | | 1238 | 19.8 CAUTION, Description changed, "CCCR.PXHD" to "M(TT)CANnCCCR.PXHD" |
| | | 1239 | 19.8 CAUTION, ID2 added |
| | | Section 25 Generic Timer Module (GTM) | |
| | | 1807 | Table 25.6 Pin Function Information, Description in the table header added, Supporting Device |
| | | Section 27 A/D Converter (ADCF) | |
| | | 2318 | 27.4.7.3 Wiring-Break Detection Diagnostic Function, Description added, NOTE |
| | | 2321 | Figure 27.16 Scan Conversion End Interrupt Occurrence Timing, Timing cycles changed |
| | | 2322 | Figure 27.17 Example of an MPX Interrupt Occurrence, Timing cycles changed |
| | | 2323 | Figure 27.18 Example of an Occurrence of A/D Error Interrupt and A/D Parity Error Notification, Timing cycles changed |
| | | Section 28 Functional Safety | |
| | | 2518 | 28.2.10.4 (1) Generating an Error Detecting Code, Description deleted |
| | | Section 35 Electrical Specifications | |
| | | 2741 | Table 35.5 Operational Condition, Description changed, Max.: For devices P1M-C (BGA-292) |
| | | 2808 | Table 35.63 Thermal Characteristics - Junction to Board Resistance (Ψjb), Description changed, Condition: P1M-C (BGA292) |
| | | Section 36 Package | |
| | | 2810 | 36.1 Package Line Up, Description added, P1M-C 156,P1H-C 156: Mass (g)[TYP] |
| | | 2812 | Figure 36.3 FBGA (156pin) outline, Figure added |
| | | Section 37 CAN Controller (MCAN) | |
| | | 2816 | Section 37 CAN Controller (MCAN), Description added |
| | | 3057 | 37.8 CAUTION, Description in the table header deleted, "R7F701327" |
| | | 3057 | 37.8 CAUTION, ID13 and ID14 added |
| | | 3059 | 37.8 CAUTION, Corrected, ID2: Description |
| | | 3060 | 37.8 CAUTION, Description added, ID5 |
| | | 3060 | 37.8 CAUTION, Description changed, ID5: Description |
| | | 3062 | 37.8 CAUTION, Description added, ID10: Description |
| | | 3062 | 37.8 CAUTION, Corrected, ID10: Description |
| | | 3062 | 37.8 CAUTION, Corrected, ID11: Description |
| | | 3063 | 37.8 CAUTION, ID13 and ID14 added |

RH850/P1x-C User's Manual: Hardware

Publication Date: Rev.0.10 Nov 05, 2014
Rev.1.30 Dec 25, 2017

Published by: Renesas Electronics Corporation

**SALES OFFICES** Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.


**Renesas Electronics America Inc.**
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

**Renesas Electronics Canada Limited**
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

**Renesas Electronics Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

**Renesas Electronics Europe GmbH**
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

**Renesas Electronics Hong Kong Limited**
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

**Renesas Electronics Taiwan Co., Ltd.**
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

**Renesas Electronics Malaysia Sdn.Bhd.**
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

**Renesas Electronics India Pvt. Ltd.**
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

**Renesas Electronics Korea Co., Ltd.**
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141



© 2017 Renesas Electronics Corporation. All rights reserved.



Colophon 4.1

# RH850/P1x-C

Renesas Electronics Corporation

R01UH0517EJ0130

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Renesas Electronics:

R7F701397AEABG-C#BC1 R7F701371EABG-C#BC1 R7F701372AEABG-C#BC1 R7F701373AEABG-C#BC1 R7F701374AEAFP-C#BA1 R7F701396AEABG-C#BC1